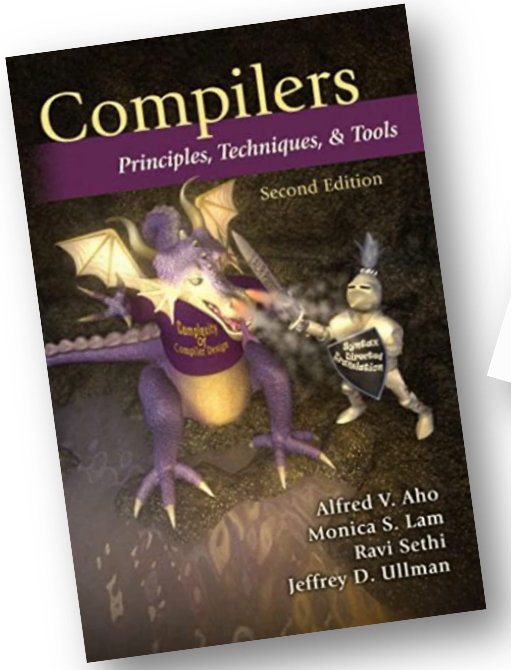
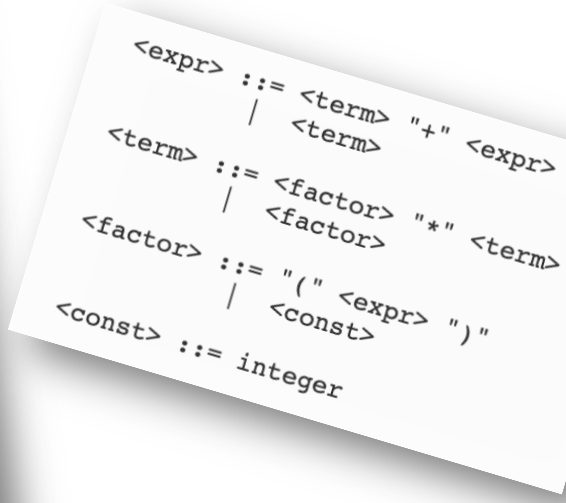
CSE211: Compiler Design

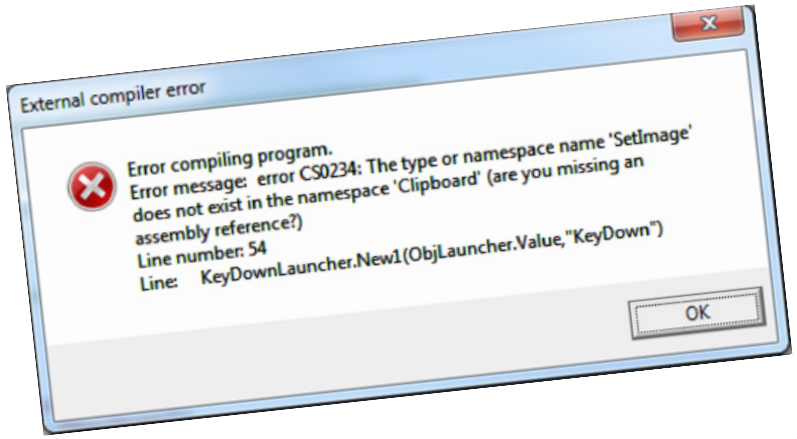
Oct. 1, 2020





- **Topic**: Course Introduction

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



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

- Tyler Sorensen

First year faculty at UCSC!

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

Introductions

- There are 13 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

Its been a tough year…

- Global Pandemic
- Graduate student strike
- Political unrest
- Wildfires

Special Note: This year has been difficult for many different reasons. I imagine that it will stay difficult for some time. Grad school is hard even in the best of times. 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. We're in this together.

Today’s class

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

- High-level discussion on compilers

Course resources

- Public course website: https://users.soe.ucsc.edu/~tsorensen/classes/cse211fall20/index.html
  - Schedule, slides, syllabus, homeworks, additional resources
- Private course website: Canvas
  - Zoom lectures/links, grades, discussions, announcements, SETs, homework solutions, tests, etc.
- Dedicated server
  - Can be used for homework. Instructions for access in Canvas

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, easily 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 (automatically) make your code more efficient 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
  - What are types and how are they used
  - how are programs structured (e.g. basic blocks)
- Comfortable using console coding (e.g. emacs or vim) and a terminal
  - Mostly Python
- Some exposure with basic parallel programming, machine code (e.g. MIPS or X86), and low-level languages (e.g. C/++).

Background

- First assignment will work as a type of gauge, especially for parsing

Modules

- There are 5 modules
- We will aim to spend roughly two weeks on each
- The last module may be cut short due to class presentations.

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 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 pointer-analysis.

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 several types of parallelization, from SIMD units available on most modern processors, to more elaborate software-hardware co-design parallelism, e.g. Decoupled Access Execute.

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 ML (e.g. tensorflow), as well as more niche languages (e.g. for graph analytics).

Modules

Module 5: Optimization Policies: This module will explore the impact of compiler optimizations and how to determine if optimizations actually provide reliable and portable performance improvements.

Class Format

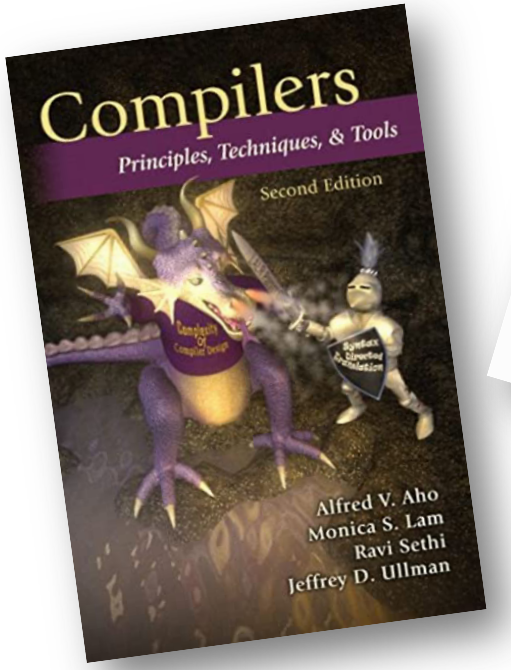
- Virtual
- However you are most comfortable
- Mute when you're not speaking
- Cameras on are nice but not required
- Zoom chat

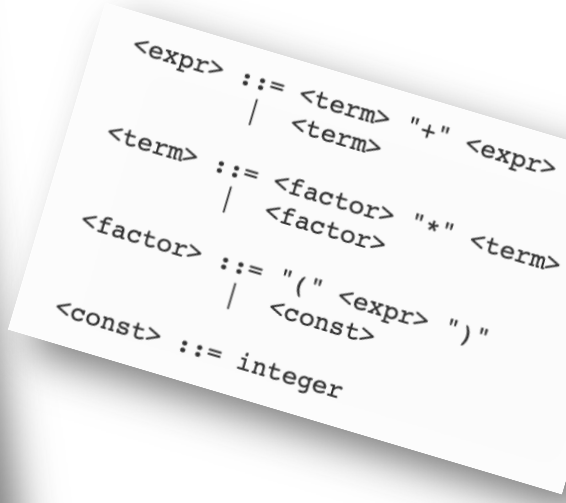
Class Format

- Class is 85 minutes
- Split into two 40 minute blocks with a 5 minute break in the middle (coffee, family, snack)
- Zoom enabled 15 minutes before/after for questions or social
- Discussion questions on start of each slide

CSE211: Compiler Design

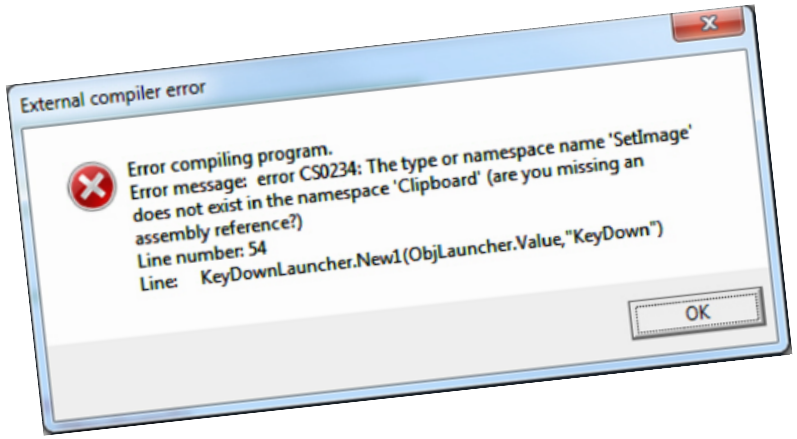
Oct. 1, 2020





- **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
  - I will moderate

- Feel free to organize outside of class (Slack, Discord)

If you do so, I only ask that you make an effort to include all your classmates and adhere to academic integrity (and also forward me any fun pictures of pets that might get posted).

Attendance

- Small class means lots of discussion! Please try your best to attend
- Message me before hand if you will miss a lecture
- 17 out of 20 lecture attendances required; more than that handled on a case-by-case basis.
- Please participate! Zoom chat, Canvas discussions, interacting with your classmates, etc.

Office Hours

- Wednesday from 3 – 4 pm Pacific Time.
  - I will open a zoom room, simply join
  - waiting room enabled, so it is private
- 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!

Homeworks

1 Homework per module, total of 5 homeworks.

- 2 weeks for each homework, posted midway through module.
- You can use the class server to work. I will install all needed software. Message me if you want additional software. Homeworks are submitted on the server as well.
- Ask questions on Canvas, discuss concepts.
- Start early!

Homeworks

Message me if you want to use languages/tools/etc. not mentioned in the homework.

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 ~90 minutes
  - No timer (I will give at least 2 days)
  - No restriction on notes and resources

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

Paper Assignment

You can exchange one of your class assignments (or midterm) with a “paper assignment”

- Approve an academic paper of interest (or I can suggest one) no later than 2 weeks before the end of classes.
- Write a 4 page double spaced review of the paper (due before the final)
- 15 minute presentation (due the last week of class)

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
- 5 page double spaced report on the project (due at the date of final)
- 15 minute presentation to the class (due last week of class)

Rubric

- 10% Attendance
- 50% Homework (10% each)
- 10% Midterm
- 30% Final

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
- Academic papers/blogs:
  - I will link to them as needed on the Resources slide

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

I will largely be using Zoom for remote lectures. You should be aware that:

- I will be recording lectures to host on Canvas. 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.
- Zoom chats are not private. Please be respectful and kind and assume everyone can see what you type.
- I will use best-practices to keep our Zoom lectures private to the class. Despite our best efforts, Zoom has a checkered history w.r.t. privacy. However, Zoom is the best tool we have available and these risks are simply part of the reality we must deal with.

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.

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/seminar/
- Topics will be highly relevant to this class; please consider joining!

Website tour

- Class website

- Canvas

Using the class server

- Email me with your desired username (maybe the same as your cruzid?) and I will send you back a password.

- Please log in ASAP and:
  - change your password. Use something secure
  - change the permission on your home directory (directions in canvas)
  - let me know if you have any issues or any software requests

- http://matt.might.net/articles/ssh-hacks/

Using the class server

- 4 core (8 thread) Intel Xeon (~3.5 GhZ). 16 GB ram
  - This is great for 1 person. Its probably about the limit for 13 people.
- Be courteous on the machine.
  - Do not do anything illegal
  - Do not saturate the cores or RAM intentionally
  - Use only for CSE211 work (unless you get permission from me)
- If you are going to depend on the machine, do not wait until the last minute.
- If the machine truly does not seem big enough, let me know and we can upgrade.

Using the class server

- Homework skeletons and testing scripts will be available on the server.
- Custom software for the homeworks will be available on the server.
- Homework must run on the server.
- However, the server is simply Ubuntu 18.04: it should be possible to develop locally, e.g. on a VM or OSX.

Using the class server

I cannot be a full-time sysadmin. If the server is abused in anyway, I will shut it down and we'll have to use the department remote unix lab.

This server should be more fun for us because we can install all sorts of interesting software!

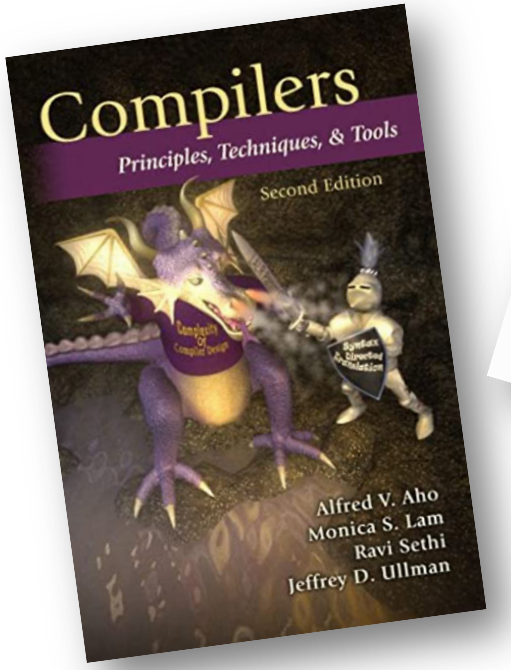
Demo

Made it through the syllabus…

The slides should start to get more interesting now

CSE211: Compiler Design

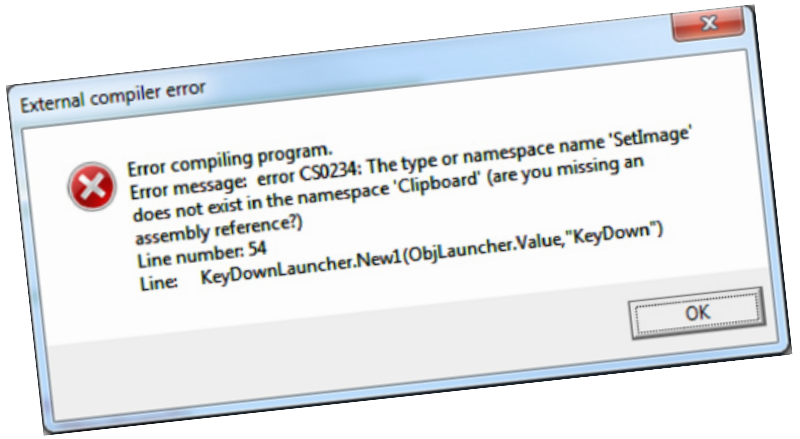
Oct. 1, 2020




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

- **Topic**: Course Introduction

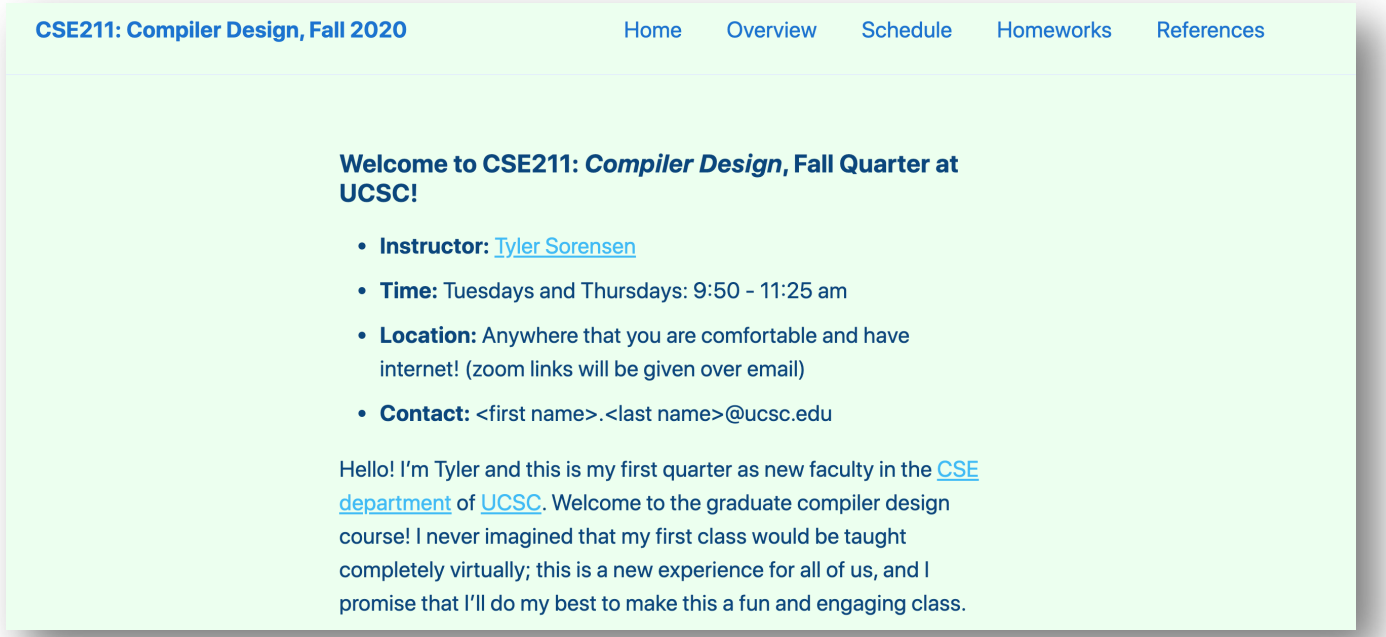
- **Questions**:
  - *What is a compiler?*
  - *What are some of your favorite compilers?*
  - *Have you ever built a compiler?*



What is a compiler?

What are some of your favorite compilers

What are some of your favorite compilers



Building this website started with:

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

What are some of your favorite compilers

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

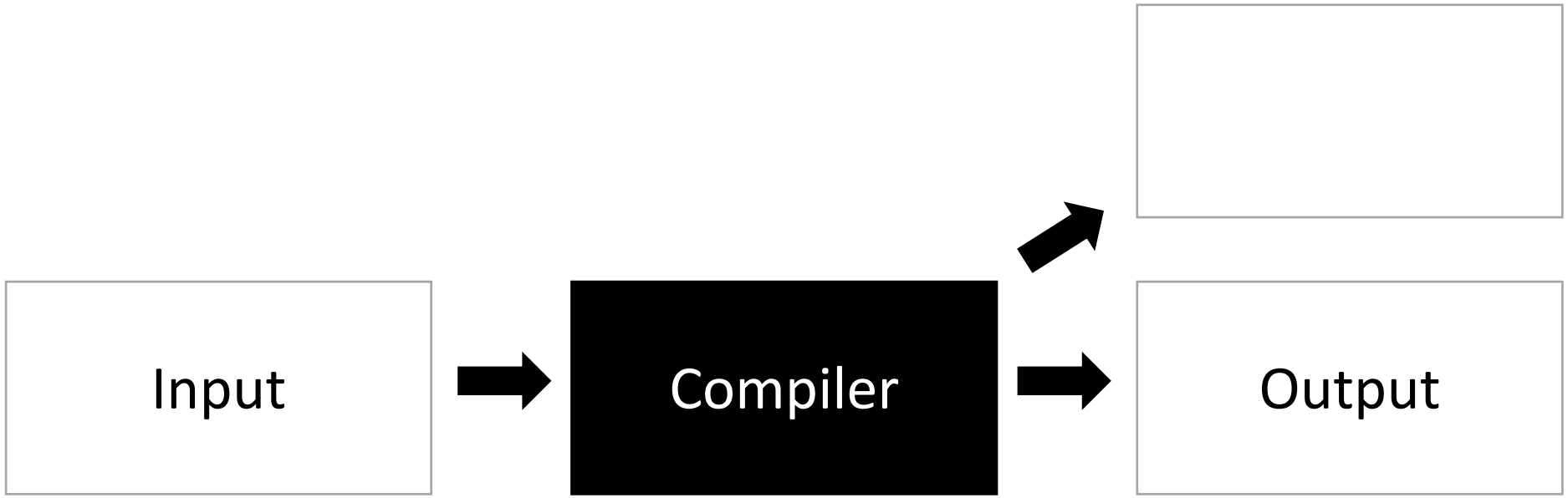
A program written in C++

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

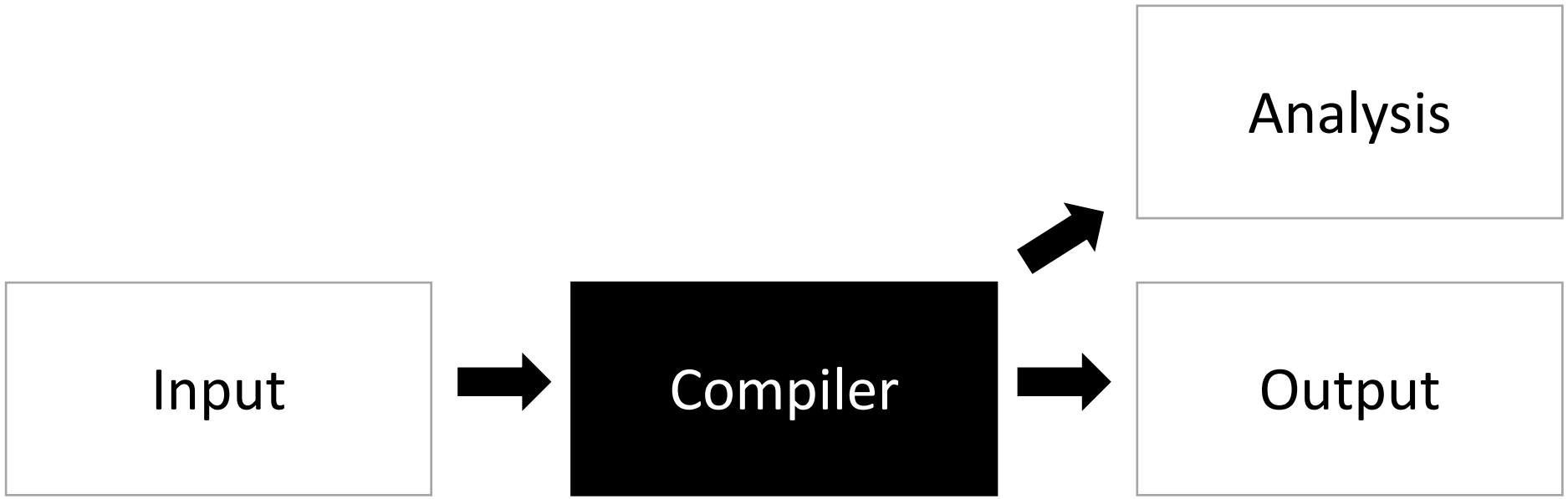
A program written in C++

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

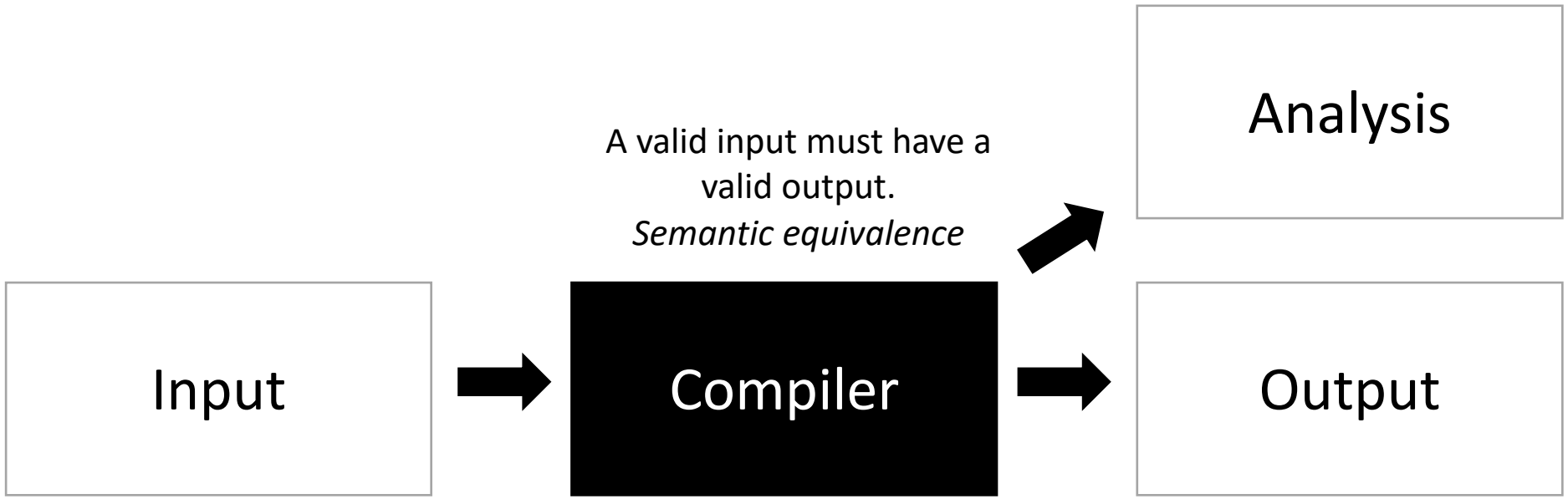
A program written in C++

Strings belonging to language L’

An executable binary file in an ISA language

An x86 Binary executable

What is a compiler?



Strings belonging to language L

A series of statements in programming language L

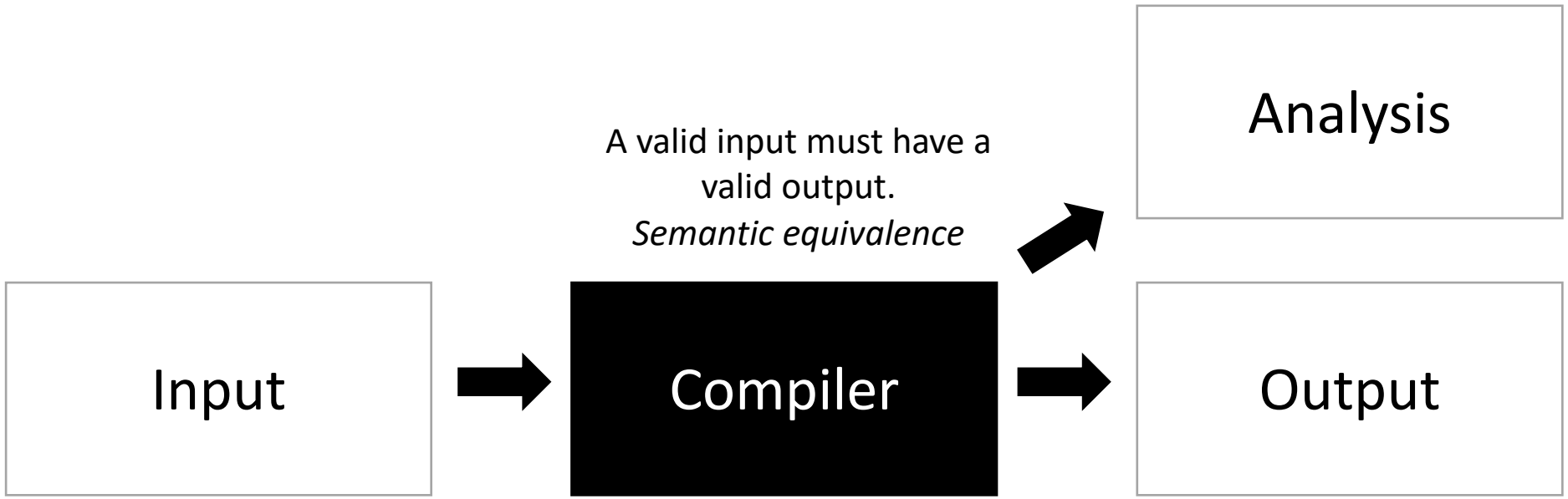
A program written in C++

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

Running this through clang on ubuntu 18.04, it returns 0. It may also segfault. It can do anything; it is undefined behavior!

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];
}
```

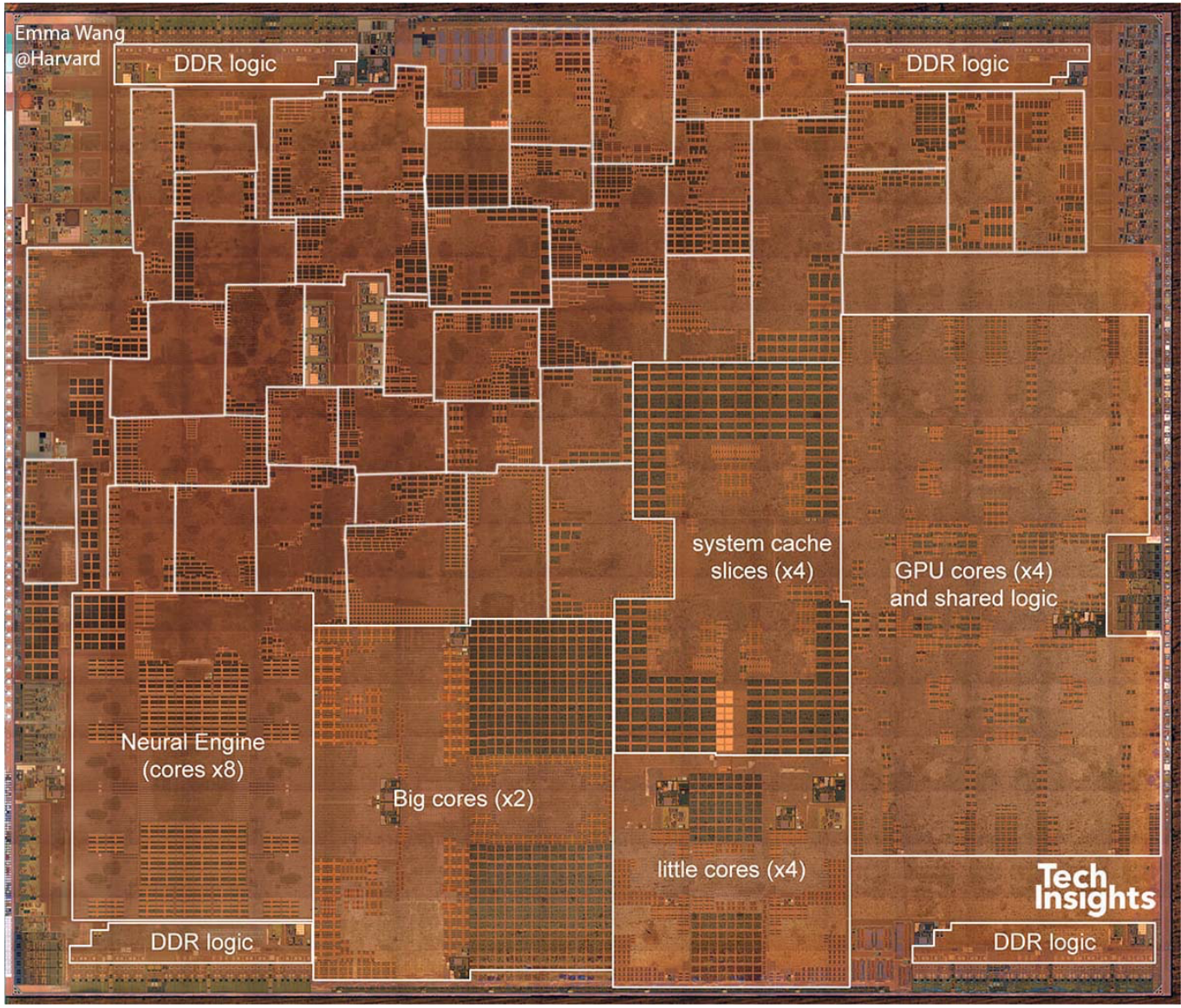
Try running this on https://godbolt.org/
change the optimization level to -O3 and see what happens!
Look for instructions like `paddd`. what does it do?

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.