CS 160 Compilers

Lecture 1: Hello World!

Yu Feng

Spring 2023

Introducing the cast

Instructor: Yu Feng yufeng@cs.ucsb.edu

Course website: https://github.com/fredfeng/CS160

Research areas: programming languages, program analysis, program synthesis, software and blockchain security

Website: http://fredfeng.github.io/

Office hour: Wed 3-4pm (HFH 2157)

Introducing the cast

TA: Junrui Liu (junrui@ucsb.edu)

Office hour: Fri TBD

TA: Thanawat Techaumnuaiwit (thanawat@umail.ucsb.edu)

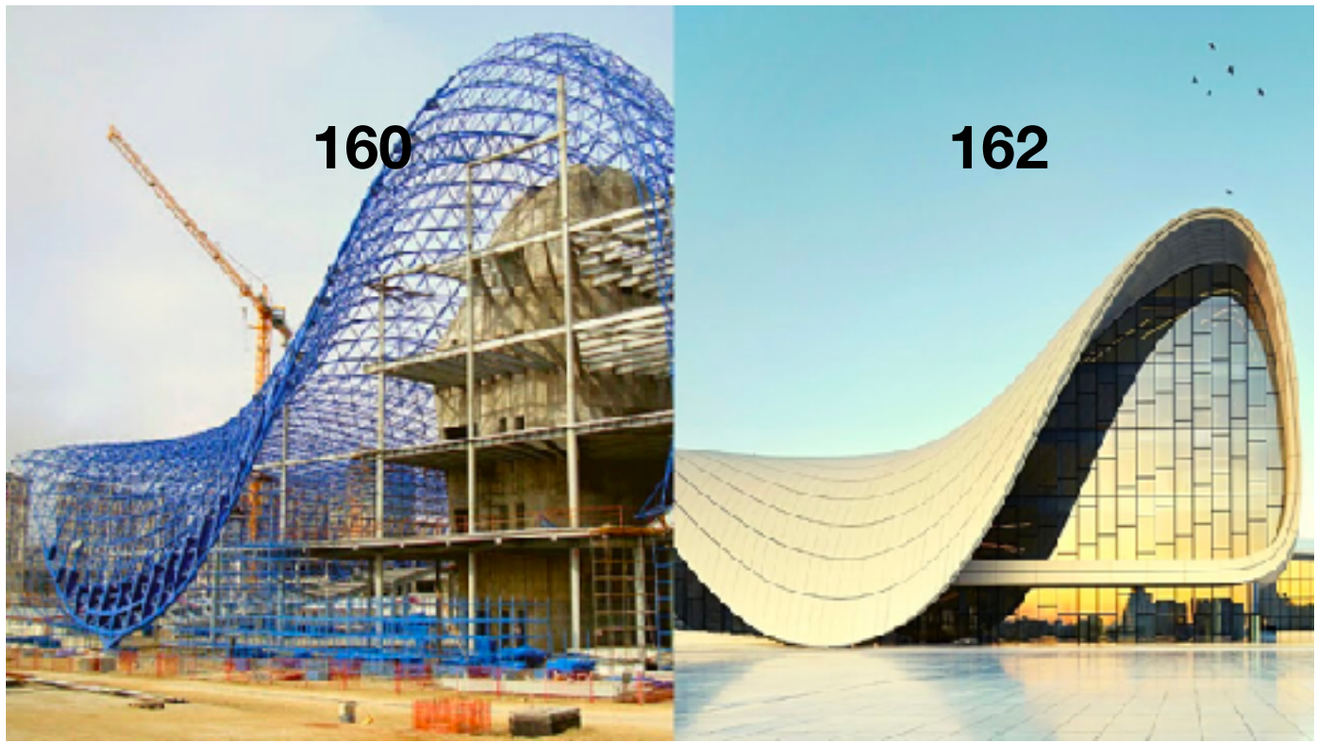
Office hour: Tue TBD

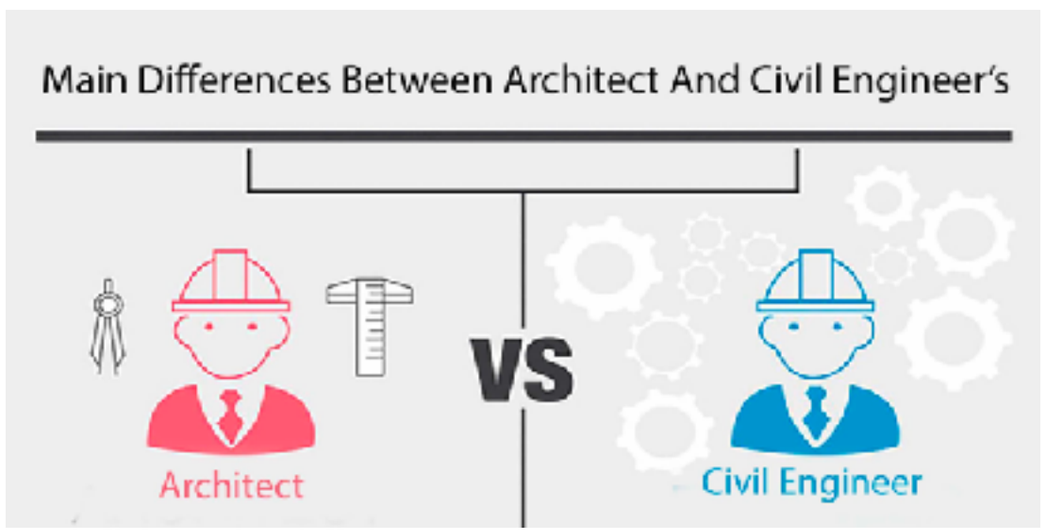
TA: Jingtao Xia (jingtaoxia@ucsb.edu)

Office hour: Thur TBD

Discussion session: Fri 9am - 11:50am

PL (CS162) v.s. Compiler (CS160)





What can we learn

You will learn:

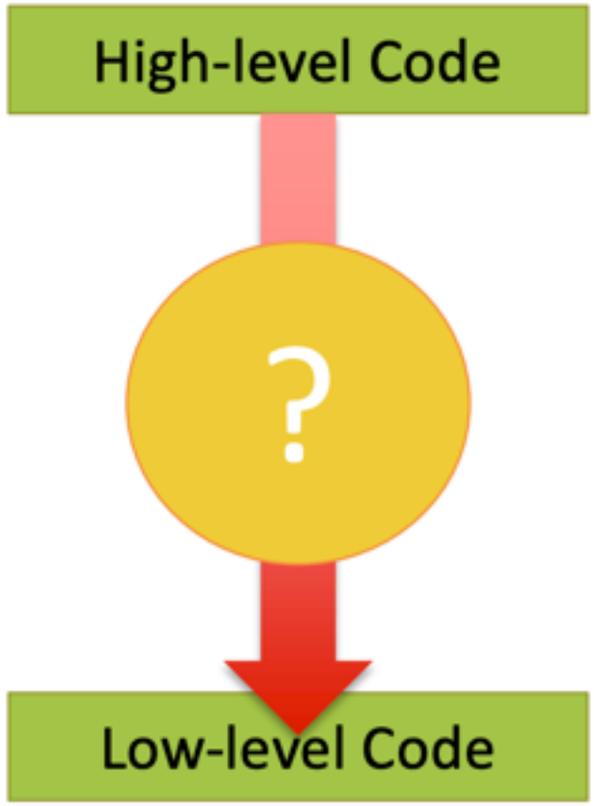
- Functional programming in OCaml
- Lexing/Parsing/Codegen
- How high-level languages are implemented in machine language
- A little about programming language semantics & types
- build a compiler (Patina) from scratch
- A deeper understanding of code: a better programmer

https://junrui-liu.github.io/patina/

What is a compiler

- A compiler is a program that translates from one programming language to another

- High-level source code to low-level machine code



History of compilers

What is the first language?

- Until the 1950's: computers were programmed in assembly.

- 1951—1952: Grace Hopper

  – developed the A-0 system for the UNIVAC I

  – Contributed significantly to the design of COBOL

- 1957: FORTRAN compiler built at IBM

- 1960's: development of the first bootstrapping compiler for LISP

- Today: thousands of languages (most little used)

Source Code

- Optimized for human readability

Why do we need compiler?

  - Expressive: matches human ideas of grammar / syntax / meaning
  - Redundant: more information than needed to help catch errors
  - Abstract: data structures, design patterns, etc.

```
#include <stdio.h>

int factorial(int n) {
  int acc = 1;
  while (n > 0) {
    acc = acc * n;
    n = n - 1;
  }
  return acc;
}

int main(int argc, char *argv[]) {
  printf("factorial(6) = %d\n", factorial(6));
}
```

Machine Code

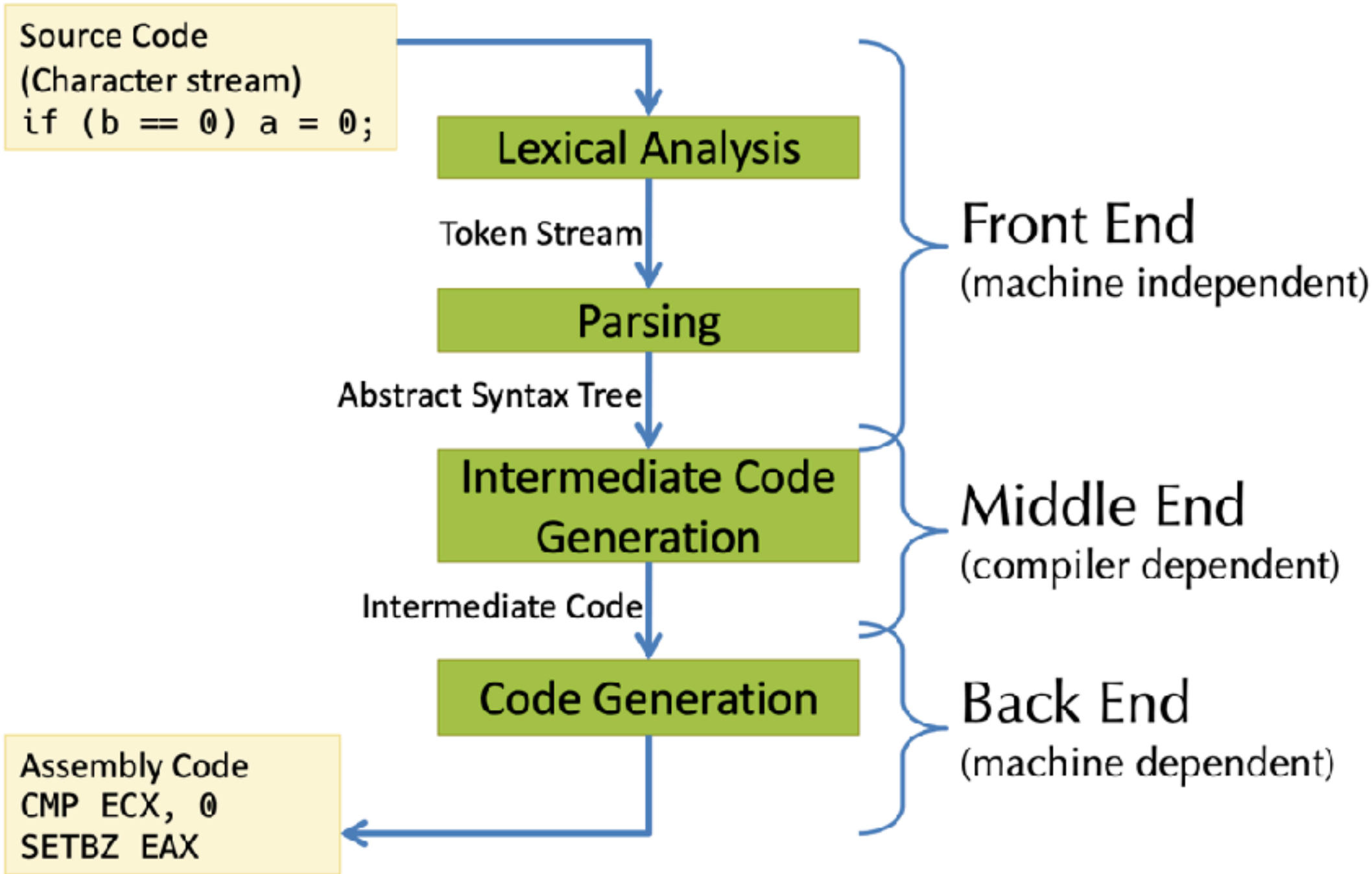
- Optimized for Hardware
  - Machine code hard for people to read
  - Reduce redundancy & ambiguity
  - Abstractions & information about intent is lost
  - Assembly language — then machine language

```
_factorial:
## BB#0:
    pushl %ebp
    movl  %esp, %ebp
    subl  $8, %esp
    movl  8(%ebp), %eax
    movl  %eax, -4(%ebp)
    movl  $1, -8(%ebp)
LBB0_1:
    cmpl  $0, -4(%ebp)
    jle   LBB0_3
## BB#2:
    movl  -8(%ebp), %eax
    imull -4(%ebp), %eax
    movl  %eax, -8(%ebp)
    movl  -4(%ebp), %eax
    subl  $1, %eax
    movl  %eax, -4(%ebp)
    jmp   LBB0_1
LBB0_3:
    movl  -8(%ebp), %eax
    addl  $8, %esp
    popl  %ebp
    retl
```

How to translate

- Mismatch between source code and machine code
- Some languages are farther from machine code than others: C++, Java, Lisp, ML, Haskell, Ruby, Python, Javascript…
- Goals of translation:
  - Source level expressiveness for the task
  - Best performance for the concrete computation
  - Reasonable translation efficiency
  - Maintainable code – Correctness!

Compiler Structure



Red or Blue

THE CHOICE IS YOURS

Grading

- Programming assignments: 100%
  - 5 (PA1-PA5) programming assignments, 20% each

Programming assignments

- Please check the website regularly
- Deadline extension:
  - Ten “late days”
  - Plan ahead, no other extensions

Programming assignments

Unfamiliar languages

\+ Unfamiliar environments

Start early!

OCaml is hard

\+ Racket is @!#@%

Start early!

Free your mind

Academic integrity

- All assignments should be done ALONE
- We use MOSS to detect plagiarism
  - Have code from public repos
  - Make sure your repo private
- “F” if you violate the honor code

Course logistics

Website: https://github.com/fredfeng/CS160

Q&A: Please join us ASAP via this QR code

TODOs by next lecture

- Join Slack for CS160!
- Install/try OCaml on your laptop
- Get familiar with your new friend: Patina manual