Introduction to Software Development Principles and Practices (SWPP)

M1522.000100

Chung-Kil Hur

(Credit: Byung-Gon Chun)

All You Ever Wanted to Know about How to Build Large-Scale Software ☺

Who am I?

- Prof. Chung-Kil (Gil) Hur [허충길]
  - Education: KAIST (B.S.), Univ. of Cambridge (Ph.D.)
  - (High school) Bronze medal in Intl' Math Olympiad (IMO) 1994.
  - Software Foundations Lab
    http://sf.snu.ac.kr
  - Research Topics
    - Software Verification (Compilers, Hypervisor, …)
    - Low-level Language Semantics (C/C++/LLVM/Rust)
    - Relaxed-Memory Concurrency
  - Our collaborators
    - Many in USA, UK, Germany, France, Israel.
  - Publications in Programming Languages (PL) area
    - POPL and PLDI are the two most prestigious conferences in PL area (see https://csrankings.org)
    - Published 20 POPL/PLDI papers (19^{th} in the history of 50 years)
    - Received distinguished paper awards from PLDI 2017, 2021 & POPL 2020

Teaching Assistants

- Instructor: Chung-Kil Hur
  - Email: gil.hur@sf.snu.ac.kr
  - Office: Bldg. 302, Rm. 426
  - Office hours: Anytime by appointment

- TAs
  - Seunghyun Nam
  - Yonghee Kim
  - Dongjae Lee
  - ChatGPT !
  - Email at swpp@sf.snu.ac.kr

- Course Web
  https://github.com/snu-sf-class/swpp202401

Goals for Today

- What is this course about?
- How does this class operate?
- Interaction is important!
  - Ask questions!

This Course is About

- Principles + Practices
  of building large-scale software systems

- An hands-on course on large-scale software systems: project-oriented
  - This semester's theme is a LLVM compiler

This Course is About

- Building large software systems that actually work is hard. This course covers techniques for dealing with the complexity of software systems

- We will focus on the technology of software development principles and software engineering for the individual and small team
  - Specifications, principles of design and software architecture, testing, abstraction, modularity, design patterns, software development process, etc.

This Course is About

- The students are expected to apply the principles to systems in practice by working on semester-long group projects on compilers

- You can think that each team is adding new functionalities to large working software. The students applies software engineering principles to build their software products.

Class Components
(subject to change)

Class participation	5%
Warm up practice (GIT, LLVM practices)	35%
Development (Documentation, Coding, Testing, Code review, …)	40%
Competition Result	20%

Course Materials

- **There is no required textbook in this class.**

- If you want to read more about the topics covered in the class, I recommend to read the following books.
  - "Engineering Software as a Service: An Agile Approach Using Cloud Computing", by Armando Fox and David Patterson
  - “Software Engineering. A Practitioner’s Approach (6th ed.) ”, by Roger Pressman
  - "Code Complete”, by Steve McConnell
  - "Design Patterns: Elements of Reusable Object-Oriented Software”, by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
  - “Extreme Software Engineering. A Hands-On Approach”, by Daniel H. Steinberg, Daniel W. Palmer
  - “Structure and Interpretation of Computer Programs (SICP) (2nd ed.)”, by Harold Abelson, Gerald Jay Sussman

  …

Course Structure

- Lecture – Tue, Thu 5:00-6:15 PM
  - Project presentations
- Practice session – Thu 7:00-9:00 PM
  - Project presentations
  - Step-by-step guidance on software development principles
- Don’t miss practice sessions: lectures and practice sessions go hand in hand

Course Timeline (subject to change)

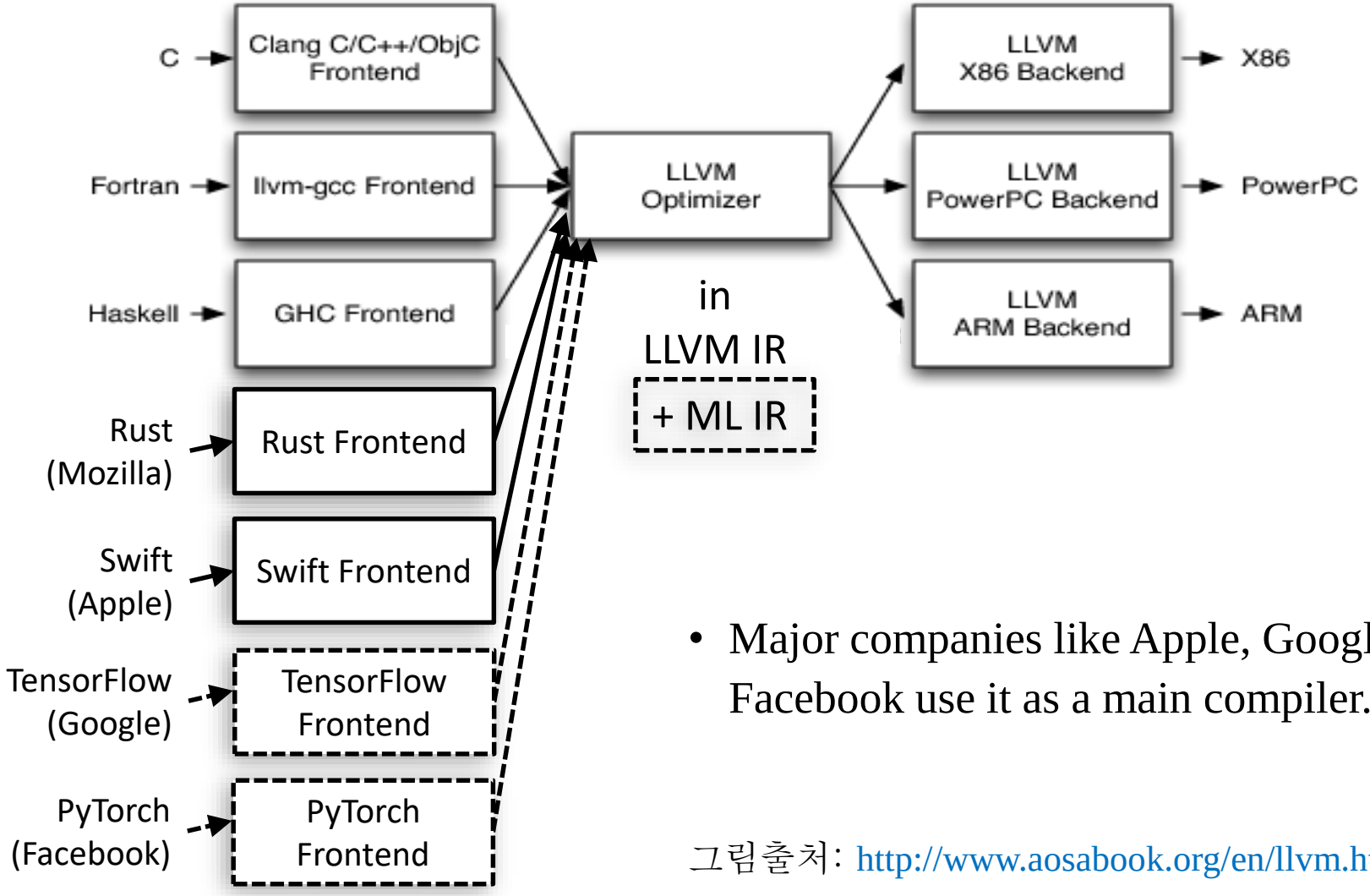
- 7 weeks
  - Learing Git & Github
  - Learning LLVM and tools
  - Individual assignments
  - Team project design, ideas and presentation
  - 1 week of team warm up
- 6 weeks
  - 3 iterations of two-week length
  - Each iteration includes:
    + Documentation: 1 day
    + Coding, Testing, Debugging: 10 days
    + Code reviews and revisions: 3 days
- 1 week
  - Wrap up and Competition

Main Project

- Goal: Develop an efficient compiler for a weird hardware using the LLVM infrastructure.

- Group: a team of 4 students (exceptionally 3)
- Start forming teams this week!

What is LLVM?



- Major companies like Apple, Google, Facebook use it as a main compiler.

그림출처: http://www.aosabook.org/en/llvm.html

Main Project, Specifically

- We provide a backend for a weird machine.
- We provide a register allocation pass.
- You develop new LLVM optimizers for LLVM IR that gain efficiency on the weird machine.

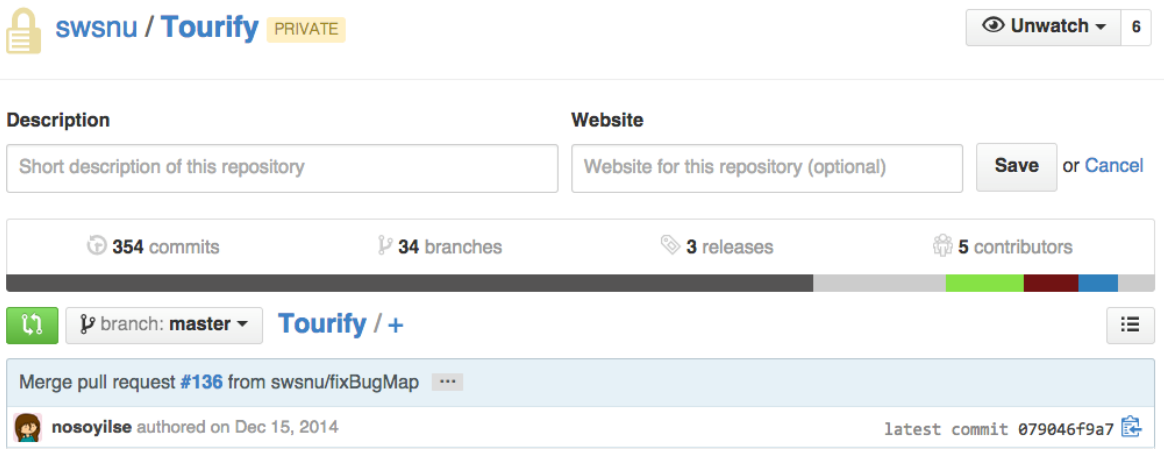
- The weird machine has sequentially accessible memory (not directly accessible) with certain magic instructions.
- We provide a weird machine simulator that computes execution cost (of time and space).

Why do we use LLVM?

- It is a large long-lived software (20 year old)
  - Can learn a lot of SE principles and practices in use (e.g., design patterns, tools, code review, …)
- Understanding existing code is one of the most common SE practices
  - You have to get used to it
  - Can learn important compiler theories (e.g., Static Single Assignment, Undefined Behavior, …)

Development

- Agile software development process
- Git for version control
- Github for project management
  - Milestones
  - Issues
  - Pull requests
  - Code review



- Testing infra – unit tests/integration tests

Timeliness

- Hard deadlines

- Catastrophic events
  - Major illness, death in family, …
  - Consult your academic advisor to come up with a plan to get back on track
  - Consult with me about this class

Cheating

- What is cheating?
  - Sharing code: by copying, retyping, looking at, or supplying a file
  - Coaching: helping your friend to write a programming assignment, line by line
  - Copying code from pervious course or from elsewhere in the Internet
  - Especially, be careful about copying code since we may open your project code! Be alert about code licenses.
- Penalty for cheating
  - F or D- & retaking this course is permanently disallowed

IMPORTANT

- Students are assumed to have C++ experience
  - If you don't, You'd better withdraw
- You should have a laptop/desktop with at least 8 GB of memory
  - If you don't, you can rent one from CS Dept for free

Welcome!
We will have lots of fun!