Compiler Construction

The Tiger Project

Context & Motivation

The needs

- 20 years ago, EPITA asked for a long and challenging problem
- Should be eventually a *potpourri* of every subject from computer science courses taught in 3^{rd} year

A (miraculous) solution

A compiler construction project!

Goals (1/3)

Complete Project

Specification, implementation, documentation, testing, distribution

Several iterations

5 (optionally up to 10) steps, 3 (resp. up to 6) months

Algorithmically Challenging

Use of well known data structure algorithms

Goals (2/3)

Team Management

Conducted in group of 4 students

Modern C++

Expressive power
Efficient
Industry standard
Low and high constructs

Object Oriented (OO) Designs

Practice common OO idioms
Apply Design Patterns

Goals (3/3)

Understanding computers

Compiler and languages are tightly related to computer architecture

Development tools

Autotools,
Doxygen,
Flex,
Bison,
ASAN ...

Secondary issue

Paradoxically

Writing a compiler is a secondary issue!

- *Why?* Only few students are likely to work in the compiler realm [Debray, 2002]
- *But...*
  - This is a unique opportunity to work on a compiler with challenging, optional assignments
  - A lot of work has to be done for GPU
  - Recruiters like this skill

Figure

- Roughly 20 years of existence
- 250/300 students per year
- Reference compiler: 25 KLOC
- Students are expected to write 5500 lines (7000 with optional assignments)
- 250+ pages of documentation

”Legacy code”

TC is the only project in which you will work in a pre-existing codebase.
This is the closest you can get to real-life industry work.

History (1/4)

2000 Beginning of the project
A front-end
A single teacher
No assistants

2001 Have students use and learn Autotools

2002 Teaching assistants involved in the project

2003 First backend, MIPS
MIPS interpretor (NoLimips)
AST and visitors are automatically generated

History (2/4)

2005 Second Teacher in the project maintainance
Use Boost
Develop Tweasts

2007 Tiger becomes an object oriented language

2009 C++ Object on the parser stack

2011 Extend Bison to support named parameters

2012 Conversion to C++11

History (3/4)

2014 More C++11/14 aiming C++17
Support ARM
Faster build system

2016 Support for LLVM

2017 Introduce Dockerfile
Move on C++17

2018 Refactor python Bindings

History (4/4)

2019 Rework assignments

2021 Move on C++20,
TC-L is now mandatory
Middle-end rework

2022 Support of Nix,
Rework of TC-1/TC-2,

2023 Switch from Flex to reFlex,
WIP: SSA, Tasks, ...

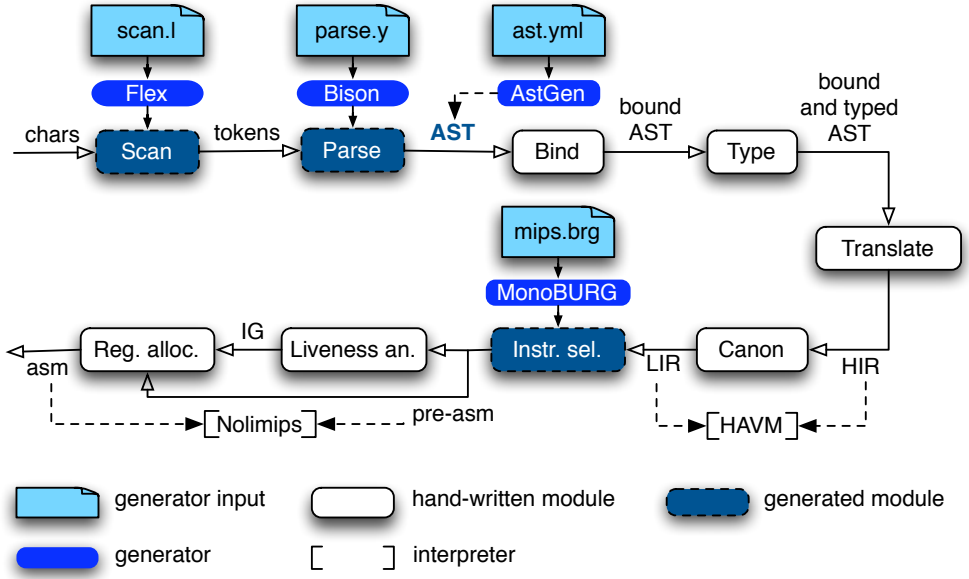
Practical informations

- Project Assignment: https://assignments.lrde.epita.fr
- Tiger Reference Manual (see link above)
- General informations (HAVM, Monoburg,...)
  https://www.lrde.epita.fr/wiki/Tiger
- Emacs Mode: https://www.lrde.epita.fr/~tiger/tc/tiger.el
- Vim Mode: https://www.lrde.epita.fr/~tiger/tc/tiger.vim

Rules of the Game

1. **No copy between groups**
2. **Tests are part of the project**
3. **Fixing mistakes earlier is better** (and less expensive)
4. **Work between group is encouraged** (as long they don't cheat)

Tiger Compiler (tc) overview



Summary