Lecture 6

CIS 341: COMPILERS

Announcements

- Colloquium Talk TODAY 3:00-4:00 in Wu & Chen

  Xuehai Qian

  Taming Relaxed Memory Consistency and Non-determinism in Parallel System

- My office hours today (only) will start at 4:00 instead of 3:30.

INTERMEDIATE REPRESENTATIONS

Eliminating Nested Expressions

- Fundamental problem:
  - Compiling complex & nested expression forms to simple operations.

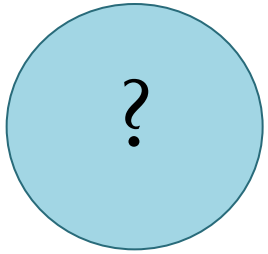
Source

```
((1 + X4) + (3 + (X1 * 5)))
```

AST

```
Add(Add(Const 1, Var X4),
    Add(Const 3, Mul(Var X1,
                     Const 5)))
```

IR



- Idea: *name* intermediate values, make order of evaluation explicit.
  - No nested operations.

Translation to SLL

- Given this:

```
Add(Add(Const 1, Var X4),
    Add(Const 3, Mul(Var X1,
                     Const 5)))
```

- Translate to this desired SLL form:

```
let tmp0 = add 1L varX4 in
let tmp1 = mul varX1 5L in
let tmp2 = add 3L tmp1 in
let tmp3 = add tmp0 tmp2 in
  ret tmp3
```

- Translation makes the order of evaluation explicit.
- Names intermediate values
- Note: introduced temporaries are never modified

See ir-by-hand, ir3.ml, ir4.ml, ir5.ml

INTERMEDIATE REPRESENTATIONS

see ll.ml in HW3

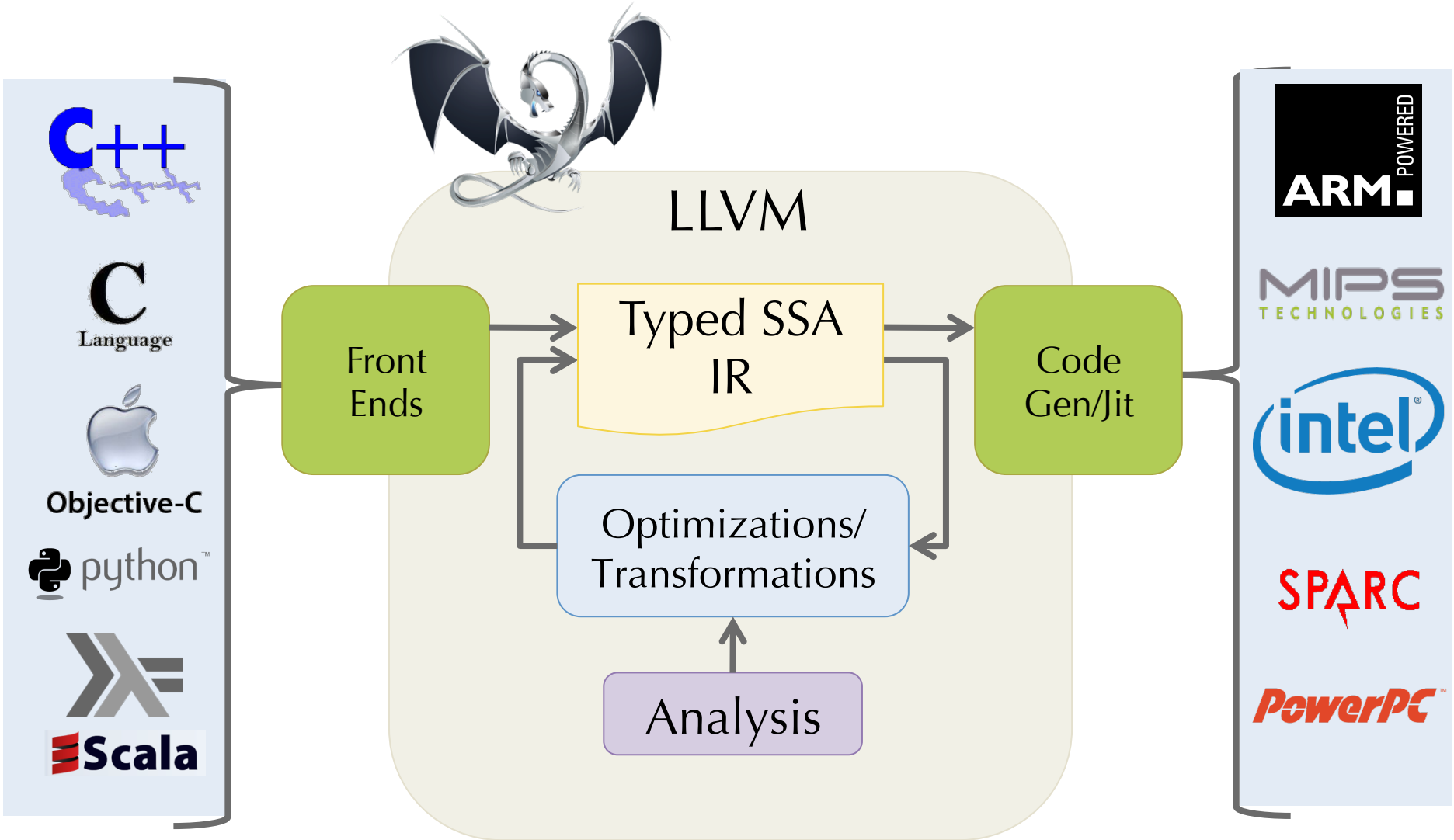
LLVM LITE

Low-Level Virtual Machine (LLVM)

- Open-Source Compiler Infrastructure
  - see llvm.org for full documntation
- Created by Chris Lattner (advised by Vikram Adve) at UIUC
  - LLVM: An infrastructure for Mult-stage Optimization, 2002
  - LLVM: A Compilation Framework for Lifelong Program Analysis and Transformation, 2004
- 2005: Adopted by Apple for XCode 3.1
- Front ends:
  - llvm-gcc (drop-in replacement for gcc)
  - Clang: C, objective C, C++ compiler supported by Apple
  - various languages: ADA, Scala, Haskell, …
- Back ends:
  - x86 / Arm / Power / etc.
- Used in many academic/research projects
  - Here at Penn: SoftBound, Vellvm

LLVM Compiler Infrastructure

[Lattner et al.]



Basic Blocks

- A sequence of instructions that is always executed starting at the first instruction and always exits at the last instruction.
  - Starts with a label that names the *entry point* of the basic block.
  - Ends with a control-flow instruction (e.g. branch or return) the "link"
  - Contains no other control-flow instructions
  - Contains no interior label used as a jump target

- Basic blocks can be arranged into a *control-flow graph*
  - Nodes are basic blocks
  - There is a directed edge from node A to node B if the control flow instruction at the end of basic block A might jump to the label of basic block B.