Lecture 6

CIS 341: COMPILERS
Announcements

• HW2: X86lite
  – Available on the course web pages.
  – Due: Thursday, February 2nd at 11:59:59pm
  – Pair-programming:
    • Register the group on the submission page
    • Submission by any group member counts for the group
INTERMEDIATE REPRESENTATIONS
Intermediate Representations

- **IR1: Expressions**
  - simple arithmetic expressions, immutable global variables

- **IR2: Commands**
  - global *mutable* variables
  - commands for update and sequencing

- **IR3: Local control flow**
  - conditional commands & while loops
  - basic blocks

- **IR4: Procedures (top-level functions)**
  - local state
  - call stack
Eliminating Nested Expressions

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

\[
\text{Source: } ((1 + X4) + (3 + (X1 \times 5)))
\]

\[
\text{AST: } \text{Add(Add(Const 1, Var X4),}
\text{ Add(Const 3, Mul(Var X1, Const 5)))}
\]

\[
\text{IR: } ?
\]

• Idea: *name* intermediate values, make order of evaluation explicit.
  – No nested operations.
Translation to SLL

- Given this:

\[
\text{Add(Add(} \text{Const 1, Var X4),} \\
\text{Add(} \text{Const 3, Mul(Var X1,} \\
\text{Const 5)}))
\]

- 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