Announcements

• HW2: X86lite
  – Due: Weds, February 12th at 11:59:59pm
  – Pair-programming:
    • Register the group on the submission page
    • Submission by any group member counts for the group
see: ir-by-hand.ml, ir<X>.ml
Eliminating Nested Expressions

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

\[
((1 + X4) + (3 + (X1 \times 5)))
\]

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

• Given this:

\[
\begin{align*}
\text{Add} & (\text{Add}(\text{Const } 1, \text{ Var } X4), \\
& \quad \text{Add}(\text{Const } 3, \text{Mul}(\text{Var } X1, \\
& \quad \quad \text{Const } 5)))
\end{align*}
\]

• Translate to this desired SLL form:

\[
\begin{align*}
\text{let } & \text{tmp0 }= \text{add } 1L \text{ varX4 in} \\
\text{let } & \text{tmp1 }= \text{mul } \text{varX1 }5L \text{ in} \\
\text{let } & \text{tmp2 }= \text{add } 3L \text{ tmp1 in} \\
\text{let } & \text{tmp3 }= \text{add } \text{tmp0 }\text{ tmp2 in} \\
& \text{tmp3}
\end{align*}
\]

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

• IR5: “almost” LLVM IR
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.