CS 473: COMPILER DESIGN
LOOPS AND DOMINATORS
Compilation in a Nutshell

Source Code
(Character stream)
if (b == 0) { a = 1; }

Token stream:
if ( b == 0 ) { a = 0 ; }

Abstract Syntax Tree:

If
  Eq
  Assn
  b
  0
  a
  1

Intermediate code:
l1:
  %cnd = icmp eq i64 %b, 0
  br i1 %cnd, label %l2, label %l3
l2:
  store i64* %a, 1
  br label %l3
l3:

Assembly Code
l1:
  cmpq %eax, %0
  jeq 12
  jmp 13
l2:
  ...

Lexical Analysis

Parsing

Analysis & Transformation

Backend
Generalizing Dataflow Analyses

• Optimization = Analysis + Transformation

• We can use dataflow analysis to figure out where an optimization applies:
  – Reaching definitions: where do we have variables with a single constant definition?
  – Available expressions: where do we compute expressions we’ve already computed?

• We can also use it to gather information for other optimizations:
  – Alias analysis: when can we store to one address without invalidating what we know about another address?
  – Loop/dominator analysis: where are the loops in the program, and how are they nested inside each other?
Loops in Control Flow Graphs

• Most programs spend most of their time in loops, so if we want to optimize, loops are a good place to look!

• Like other optimizations, we want to apply loop optimizations to a control flow graph IR

• Other opts may create opportunities for loop opts and vice versa, so it makes sense to alternate between them

• Loops may be hard to recognize at the CFG IR level
  – Many kinds of loops: while, do/while, for, loops with break/continue, goto... all of which turn into some combination of comparisons, labels, jumps, and body code

• Goal: find all the sections of a CFG that can be treated as loops
Definition of a Loop

- Exercise: Which of these nodes form a loop? Where does the loop start, and where does it end?
Definition of a Loop

• Exercise: Which of these nodes form a loop? Where does the loop start, and where does it end?

• Every loop has a single entry point, called the *header*
Definition of a Loop

- Exercise: Which of these nodes form a loop? Where does the loop start, and where does it end?

- Every loop has a single entry point, called the header

- Every node in the loop is reachable from the header, and the header is reachable from every node
  - A loop is a strongly connected component

- We can only get into the loop from outside it through the header

- We can write the loop as 2: 1, 3, 4, 6
Definition of a Loop

• Exercise: Which of these nodes form a loop? Where does the loop start, and where does it end?

• Every loop has a single entry point, called the *header*

• Every node in the loop is reachable from the header, and the header is reachable from every node
  – A loop is a *strongly connected component*

• We can only get into the loop from outside it through the header

• Nodes with outgoing edges are called *exit nodes* for the loop
Questions

Top
Nested Loops

- Control-flow graphs may contain many loops
- Loops may contain other loops:

- Goal: Build a loop nesting tree for the CFG, so we know where to focus our optimizations
• Every loop has a single entry point, called the **header**

• Every node in the loop is reachable from the header, and the header is reachable from every node
  – A loop is a *strongly connected component*

• We can only get into the loop from outside it through the header

• Nodes with outgoing edges are called **exit nodes** for the loop
Loop Analysis

- Goal: identify the loops and nesting structure of the CFG
- Basic idea: a loop has a starting point and a body, and we can get from any node in the body back to the starting point
- The hard part is finding possible starting points!
Loop Analysis

- Goal: identify the loops and nesting structure of the CFG
- The hard part is finding possible starting points!

- The header of a loop should be its *only* entry point: we can only get to any part of the loop body if we go through the header first
- So we can find headers by calculating *dominators*: node A *dominates* node B if the only way to reach B from the start of the graph is by going through A

- The header of a loop dominates all the nodes in the loop
Dominator Dataflow Analysis

- We can find the dominators of every node in a graph using our usual dataflow analysis framework.
- In fact, it’s even simpler than usual, since we only care about edges, not the statements in nodes!
  \[ \text{gen}[n] = \{n\} \text{ since every node dominates itself} \quad \text{kill}[n] = \emptyset \]

- \( \text{out}[n] := \text{in}[n] \cup \{n\} \)
- \( \text{in}[n] := \bigcap_{n' \in \text{pred}[n]} \text{out}[n'] \): node B is dominated by node A only if A dominates all of the predecessors of B

- We start with every node in all of the sets, and in each iteration remove those that don’t dominate all of the node’s predecessors
- At the end, \( \text{out}[n] \) will be the set of nodes that dominate n
Questions

Top
Dominator Trees

• Result: for each node, a set of nodes that dominate it
• Each node has one immediate dominator, the closest node that dominates it
• We can draw a tree of all the nodes in the CFG, where each node’s parent is its immediate dominator

CFG:
Dominator Trees

- Result: for each node, a set of nodes that dominate it
- Each node has one *immediate dominator*, the closest node that dominates it
- We can draw a tree of all the nodes in the CFG, where each node’s parent is its immediate dominator
**Dominator Trees**

- Result: for each node, a set of nodes that dominate it

**CFG:**

![CFG Diagram]

**Dominator Tree:**

![Dominator Tree Diagram]
Dominator Trees

• Result: for each node, a set of nodes that dominate it
• Edges that go up the dominator tree are *back edges*
• And every back edge is part of a loop!

**CFG:**

```
  1
 / \
2   3
 /   |
4     5
 /     |
6       7
 /         |
8           9
 /             |
0               0
```

**Dominator Tree:**

```
  1
/   \
2     3
/     |
4       5
 /     |
6       7
 /         |
8           9
 /             |
0               0
```
Completing Loop Analysis

- Dominator analysis identifies **back edges**: edges $n \rightarrow h$ where $h$ dominates $n$
- Each back edge has a **natural loop**:
  - $h$ is the header
  - All nodes reachable from $h$ that also reach $n$ without going through $h$ are in the loop
Completing Loop Analysis

- Dominator analysis identifies *back edges*: edges $n \rightarrow h$ where $h$ dominates $n$
- Each back edge has a *natural loop*:
  - $h$ is the header
  - All nodes reachable from $h$ that also reach $n$ without going through $h$ are in the loop
- It’s convenient for each loop to have a unique header, so if two loops share the same header, we merge them
- If all the nodes in one loop are also in another loop, the first loop is *nested* inside the second
Example Natural Loops

Natural Loops

Loop Nesting Tree:
Questions
Using Loop Information

• Loop nesting depth plays an important role in optimization heuristics
  – Optimizations pay off more in deeply nested loops

• Need to know loop headers/back edges for:
  – loop invariant code motion (remove code from loop if it’s the same every time)
  – loop unrolling (execute n iterations of the loop at once)
  – and lots more!

• Dominance information also plays a role in converting to Static Single Assignment form, another useful IR