Lecture 18
Global Scheduling

Reading: Chapter 10.4

Introduction to Global Scheduling
Assume each clock can execute 2 operations of any kind.

```
if (a==0) goto L
LD R6 <- 0(R1)  
EQZ  R6, L
LD R7 <- 0(R2)  
LD R8 <- 0(R4)  
ADD R8 <- R8,R8
ST 0(R5) <- R8
```

Result of Code Scheduling
```
LD R6 <- 0(R1)  ; LD R8 <- 0(R4)  
LD R7 <- 0(R2)  ; ADD R8 <- R8,R8
ST 0(R5) <- R8
```

Terminology

Control equivalence:
- Two operations $o_1$ and $o_2$ are control equivalent if $o_1$ is executed if and only if $o_2$ is executed.

Control dependence:
- An op $o_2$ is control dependent on op $o_1$ if the execution of $o_2$ depends on the outcome of $o_1$.

Speculation:
- An operation $o$ is speculatively executed if it is executed before all the operations it depends on (control-wise) have been executed.

Requirements:
- does not raise an exception
- satisfies data dependences
Code Motions

Goal: Shorten execution time probabilistically

Moving instructions up:
• Move instruction to a cut set (from entry)
• Speculation: even when not anticipated.

Moving instructions down:
• Move instruction to a cut set (from exit)
• May execute extra instruction
• Can duplicate code

General-Purpose Applications

• Lots of data dependences
• Key performance factor: memory latencies
• Move memory fetches up
  – Speculative memory fetches can be expensive
• Control-intensive: get execution profile
  – Static estimation
    • Innermost loops are frequently executed
      – back edges are likely to be taken
    • Edges that branch to exit and exception routines are not likely to be taken
  – Dynamic profiling
    • Instrument code and measure using representative data

A Note on Data Dependences

a = 0
\[ a = 1 \]

A Basic Global Scheduling Algorithm

• Schedule innermost loops first
• Only upward code motion
• No creation of copies
• Only one level of speculation
Program Representation

- **A region** in a control flow graph is:
  - a set of basic blocks and all the edges connecting these blocks,
  - such that control from outside the region must enter through a
    single entry block.

- **A function** is represented as a hierarchy of regions
  - The whole control flow graph is a region
  - Each natural loop in the flow graph is a region
  - Natural loops are hierarchically nested

- **Schedule regions from inner to outer**
  - treat inner loop as a black box unit
    - can schedule around it but not into it
  - ignore all the loop back edges → get an acyclic graph

Algorithm

Compute data dependences;
For each region from inner to outer {
  For each basic block B in prioritized topological order {
    \[
    \text{CandBlocks} = \text{ControlEquiv}(B) \cup
    \text{Dominated-Successors}(\text{ControlEquiv}(B));
    \]
    \[
    \text{CandInsts} = \text{ready operations in CandBlocks};
    \]
    For (t = 0, 1, ... until all operations from B are scheduled) {
      \[
      \text{S}(n) = \langle B, t \rangle
      \]
      Update resource commitments
      Update data dependences
    }
  }
  Update CandInsts;
}

Priority functions: non-speculative before speculative

Extensions

- Prepass before scheduling: **loop unrolling**
- Especially important to move operation up loop back edges

Summary

- **Global scheduling**
  - Legal code motions
  - Heuristics