

# Porting OpenIMPACT Technology to GCC

Robert Kidd  
Gelato Conference  
April 24, 2006

# Using OpenIMPACT to enhance GCC

---

- OpenIMPACT: Aggressive research compiler targeting IA64 exclusively
- GCC: Production quality compiler targeting wide range of platforms
- Want to apply lessons from OpenIMPACT to GCC

# How do GCC and OpenIMPACT compare?

- On paper, OpenIMPACT and GCC mainline support similar optimizations
  - Superblock formation
  - Inlining
  - Speculation (thanks to RAS-ISP)
  - Classical optimizations
- GCC has improved, but still lags OpenIMPACT
  - OpenIMPACT ~1.5x GCC 3.2
  - OpenIMPACT ~1.2x GCC mainline

# Accounting for the difference

- What factors contribute to the difference in performance with similar opti?
  - Optimizers are less aggressive
    - Correctness issues
    - Performance across many architectures
    - Compile-time performance
  - Completeness
    - Alias information is still a work in progress
  - Phase ordering

# Which factor can we attack?

- **Optimizer level of aggression**
  - Ensuring correctness may be difficult
  - May not be appropriate for all architectures
  - May adversely affect compiler run time
- **Completeness**
  - Project to improve alias analysis well underway
- **Phase ordering**
  - Adjustments to existing parts of GCC
  - Can approximate more aggressive optimization
  - Somewhat easier to check correctness
  - This is our focus

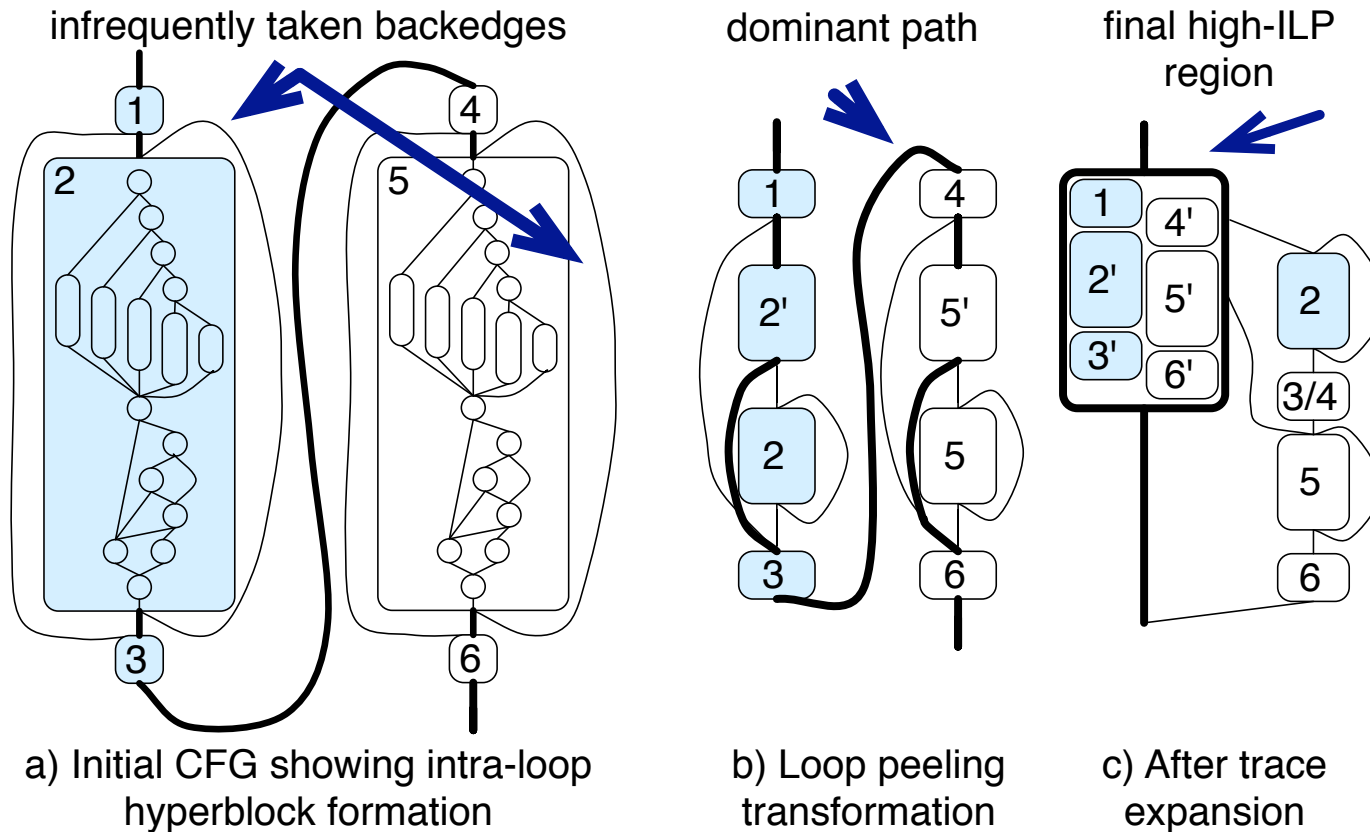
# Superblock formation in GCC

---

- Superblock formation is done late, just before instruction scheduling
- High level optimizers do not have an opportunity to specialize the modified control flow graph
- Does not achieve the full benefit of Superblock-based optimization

# Structural Transformation in OpenIMPACT

- Transform control flow graph early, allow optimizers to specialize the result



Example from 186.crafty [SiasISCA04]

# 186.crafty in OpenIMPACT

---

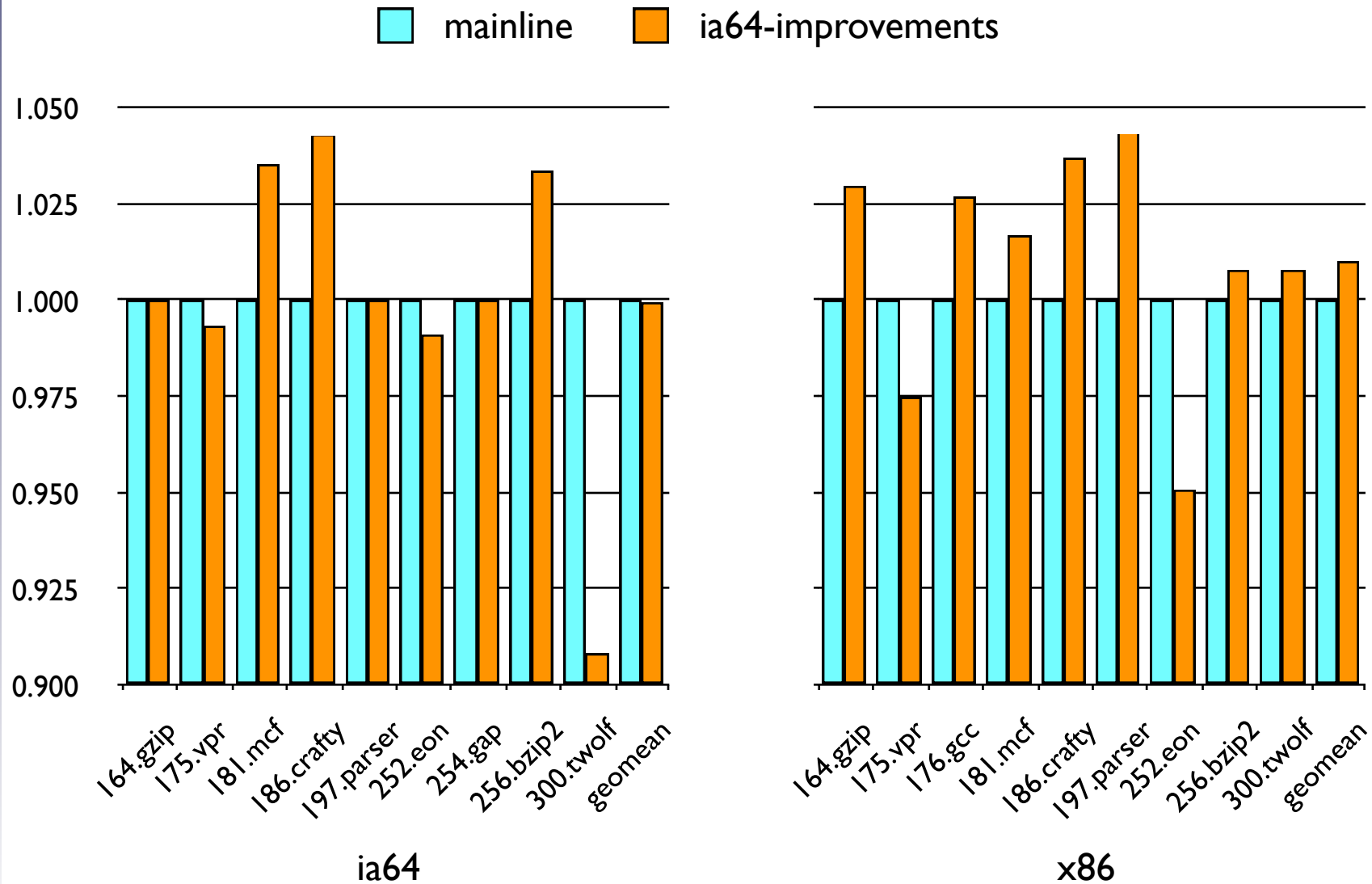
- Structural techniques transform CFG
- Traditional optimizations specialize the duplicated code
- Inside the expected path of execution, more ILP is developed in a shorter schedule
- ~9% improvement

# Apply these techniques to GCC

---

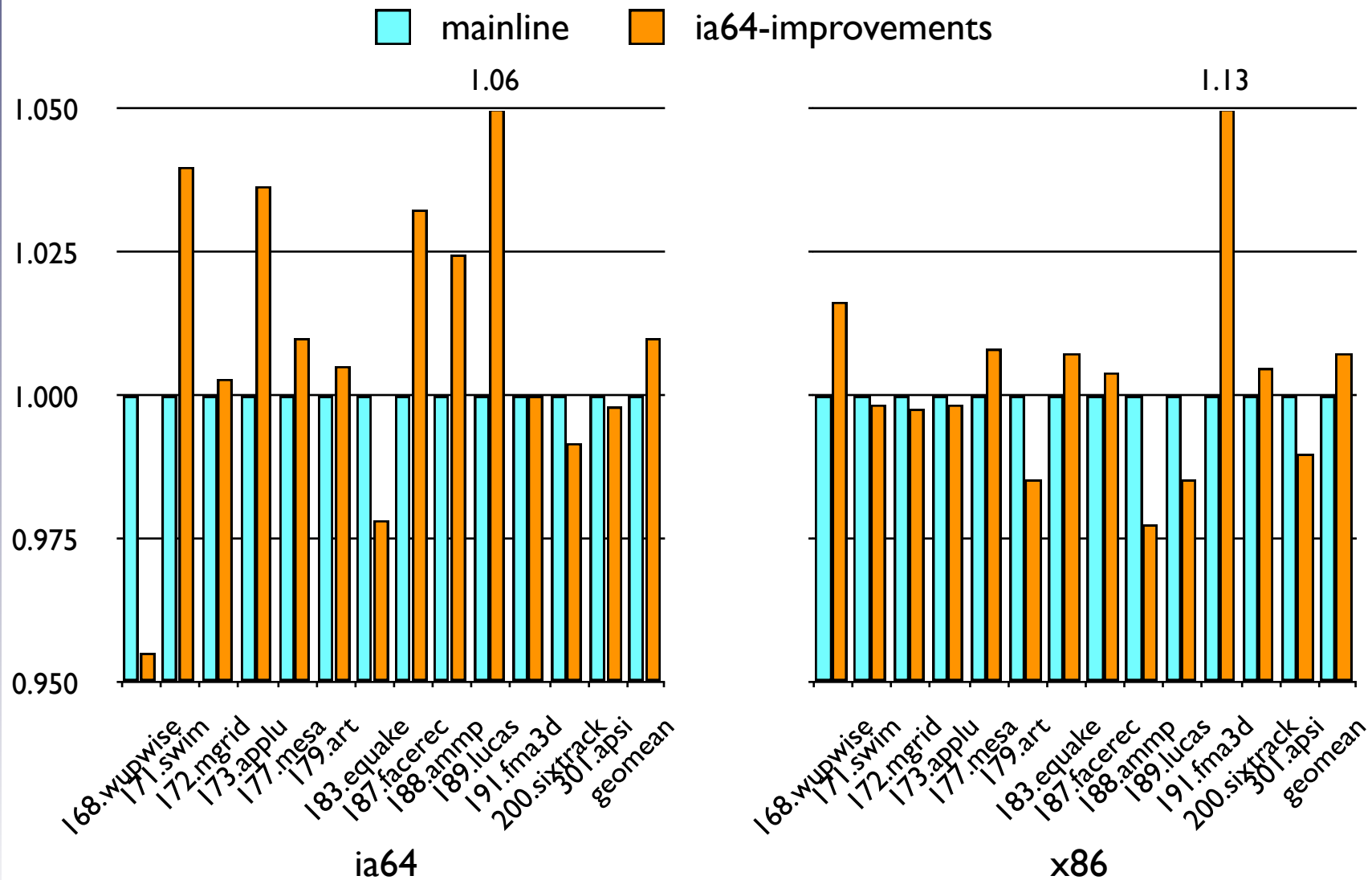
- Moved Superblock formation pass to Tree-SSA level
- Perform loop header duplication during Superblock pass to form loops that can be processed by GCC's loop optimizers
- Adjust code expansion limits to allow more code duplication
- Work is available in the ia64-improvements branch in the GCC SVN repository

# Results – Integer



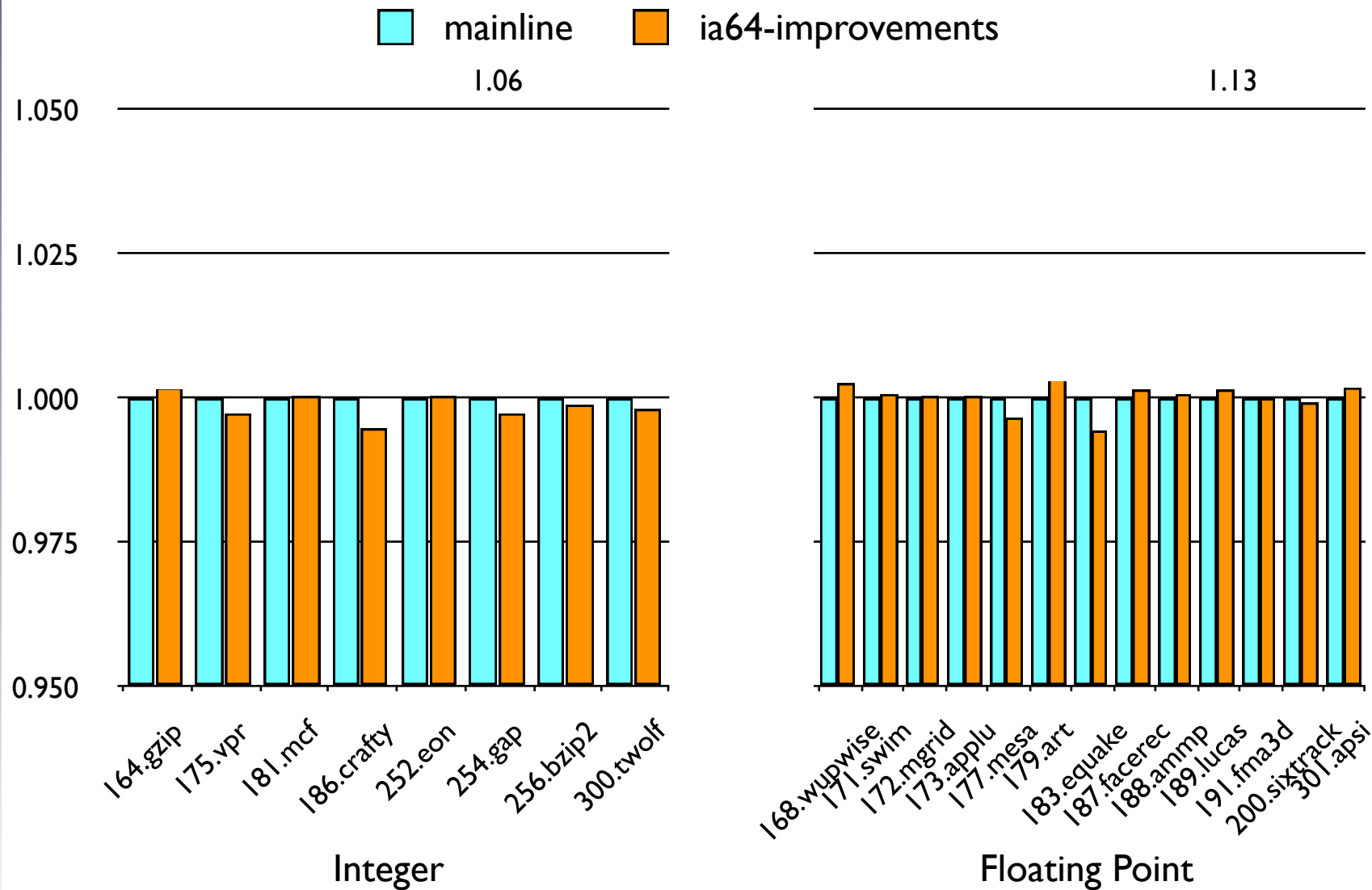
Estimated SPECint2000 speedup over GCC mainline

# Results – Floating Point



Estimated SPECfp2000 speedup over GCC mainline

# Results – Executable Size



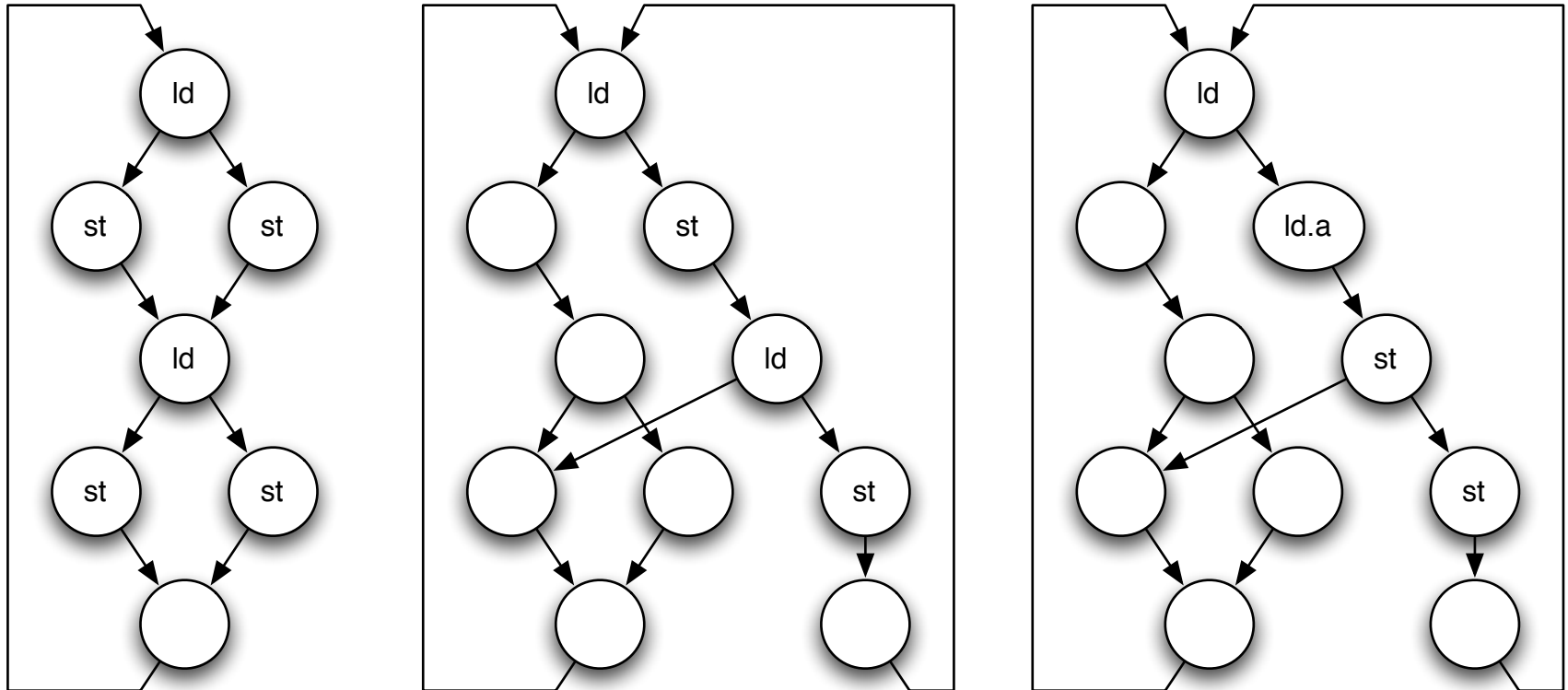
IA64 executable size relative to GCC mainline

# Analysis: 186.crafty

- 186.crafty improves by 4% when Superblocks are formed early

|                               | Mainline | ia64-improvements |
|-------------------------------|----------|-------------------|
| Useful instructions per cycle | 1.161    | 1.176             |
| Total stalls                  | 0.495    | 0.467             |
| Branch mispredictions         | 19.3e9   | 18.3e9            |
| L1I hit rate                  | 0.808    | 0.809             |
| L2I hit rate                  | 0.997    | 0.998             |

# GCC Optimization on Superblocks



# Issues still to be resolved

---

- Loop header duplication doesn't happen in some cases
- Superblock formation may adversely affect alias analysis
  - These two issues are likely the cause of the degradation in 300.twolf

# Future work and conclusion

---

- Ensure Superblocks are formed in a consistent manner
- Investigate interaction of Superblocks and other opti
- Other early Superblock-like opti
  
- Structural transformation is a proven method to develop ILP in control intensive code
- Beginning to demonstrate this in GCC