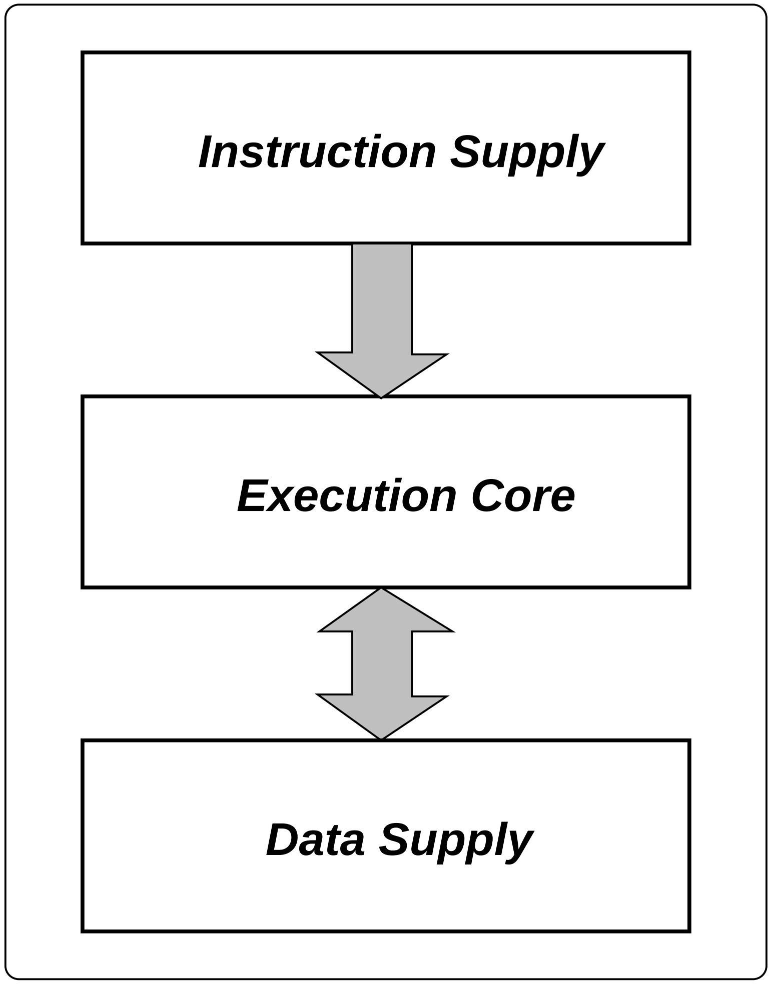
Research Objective:

Develop an instruction delivery mechanism for a 16-wide machine

Outline

- Processor Performance and the Partial Fetch Problem
- Our Solution: The Trace Cache
- Other Solutions
- Performance
- Analysis of Performance
- Summary

High Performance Processor Design



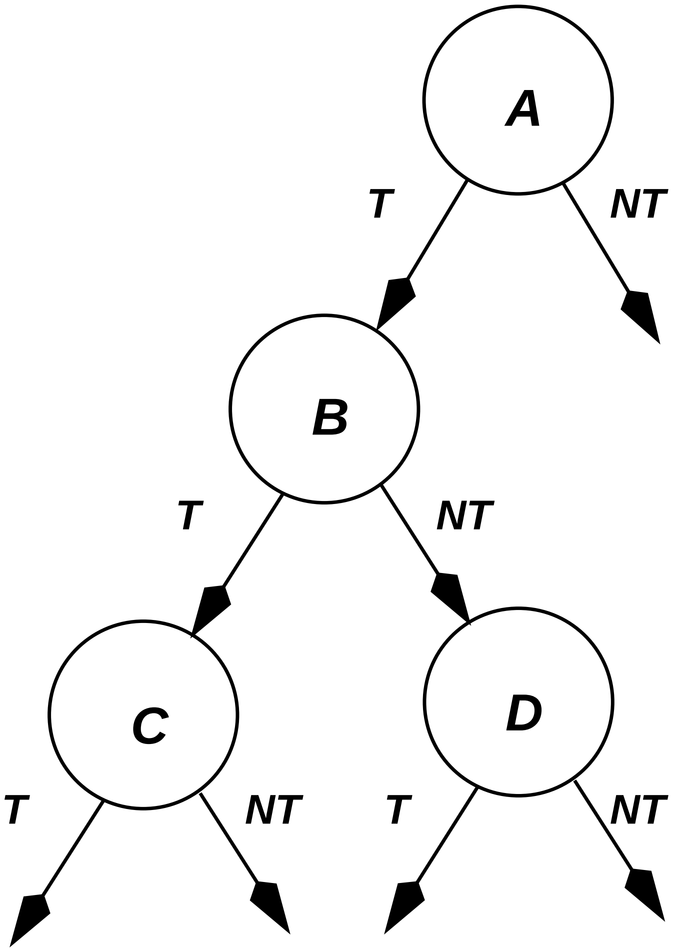
Maximizing Instruction Bandwidth

- No empty fetches (caused by cache misses)
- No fetches thrown away (caused by mispredictions)
- No partial fetches (caused by branches)

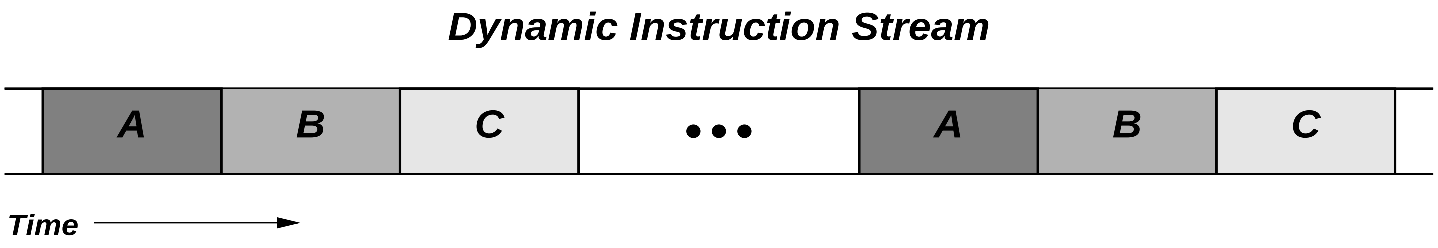
Average block lengths

	Inst/branch	Inst/taken branch
compress	9.15	13.07
gcc	5.34	8.46
go	6.21	9.11
ijpeg	15.59	19.14
li	5.54	8.41
m88ksim	4.24	5.78
perl	6.06	9.13
vortex	6.57	11.05
Average	7.34	10.52

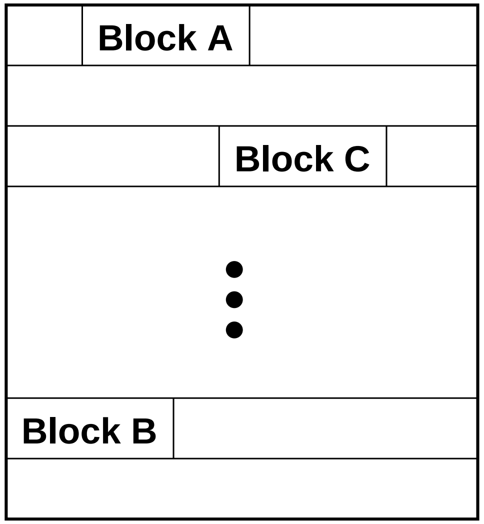
Our Solution: The Trace Cache


A
T
NT
B
T
NT
C
D
T
NT
T
NT

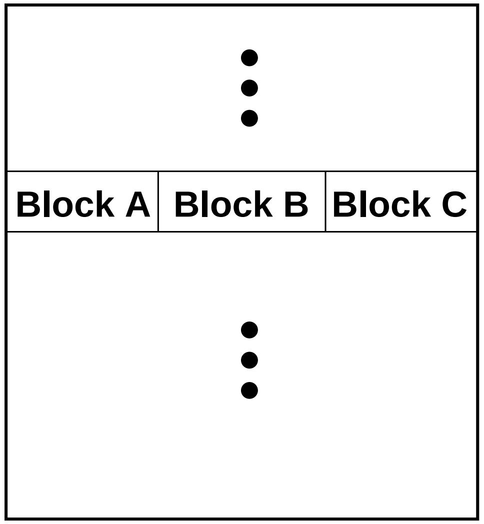
Caching the dynamic instruction stream



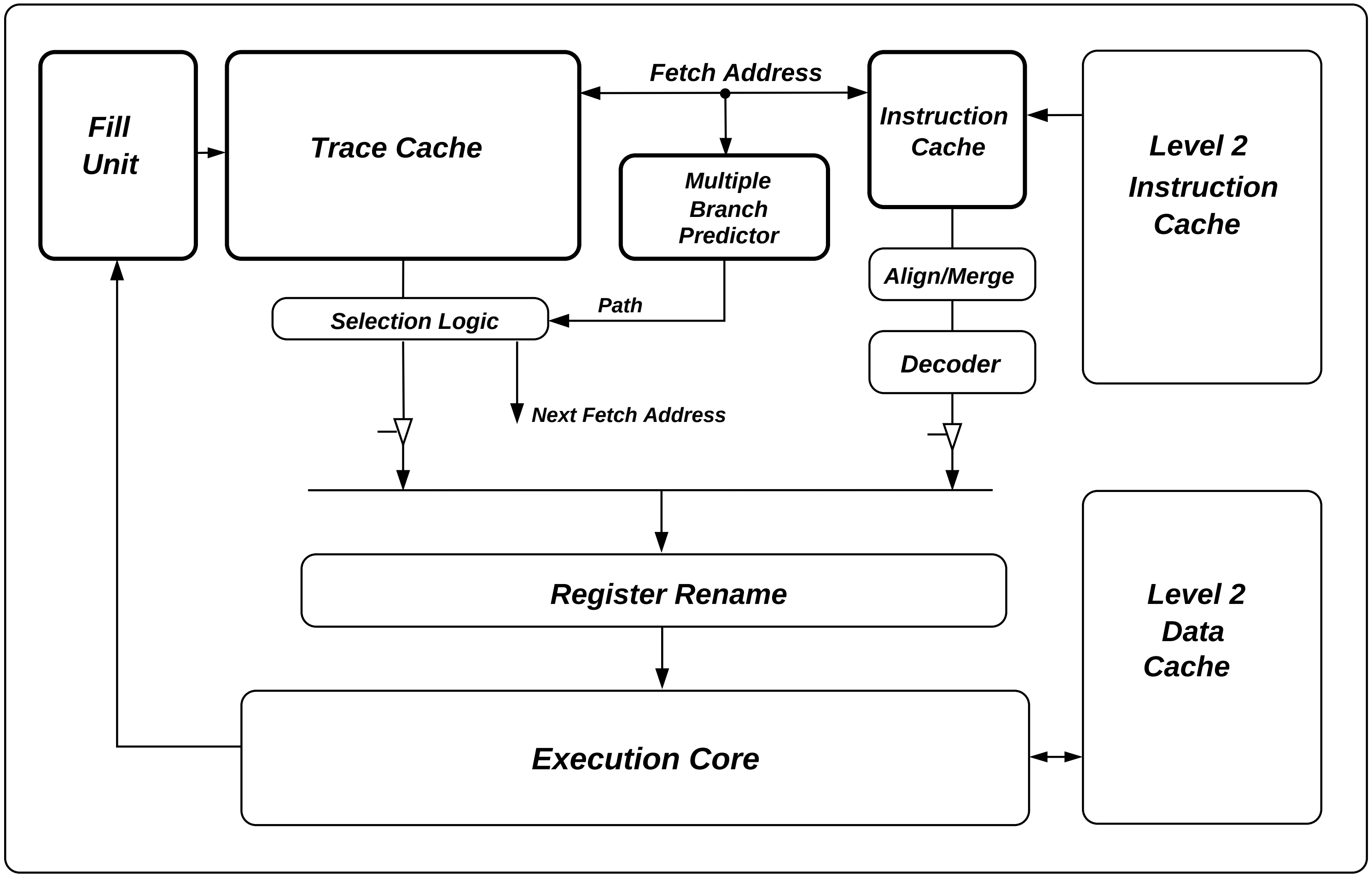
Instruction Cache



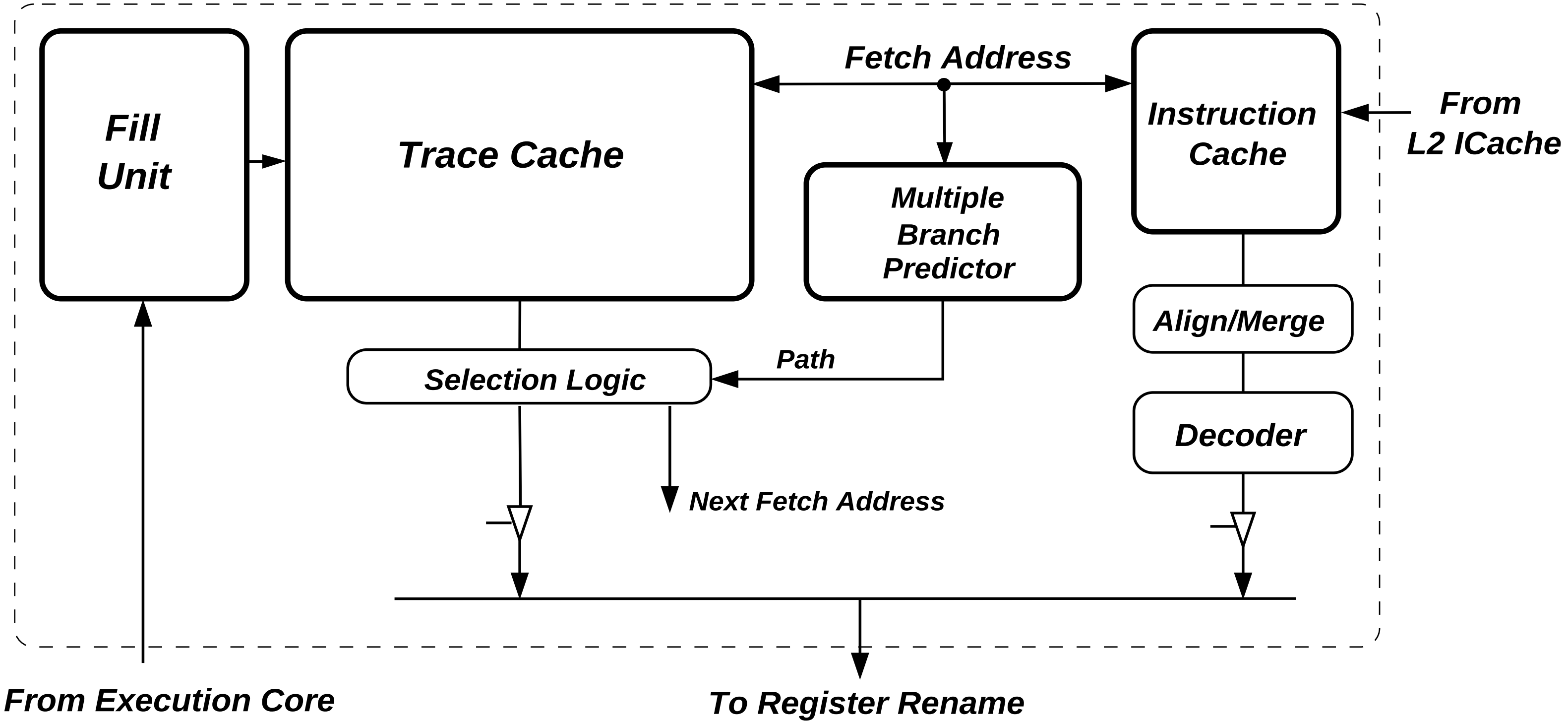
Trace Cache



The Microarchitecture



Trace Cache Fetch Mechanism



Other Solutions

Other approaches

- Hardware techniques
  - Branch Address Cache - Yeh et al., 1993
  - Collapsing Buffer - Conte et al., 1993
  - Multiple-Block Ahead - Seznec et al., 1996
- Compiler techniques
  - Trace Scheduling - Fisher, 1981
  - Superblock, Hyperblock, Extended Basic Blocks
  - BS-ISA, block enlargement - Hao et al., 1997

Trace Cache history

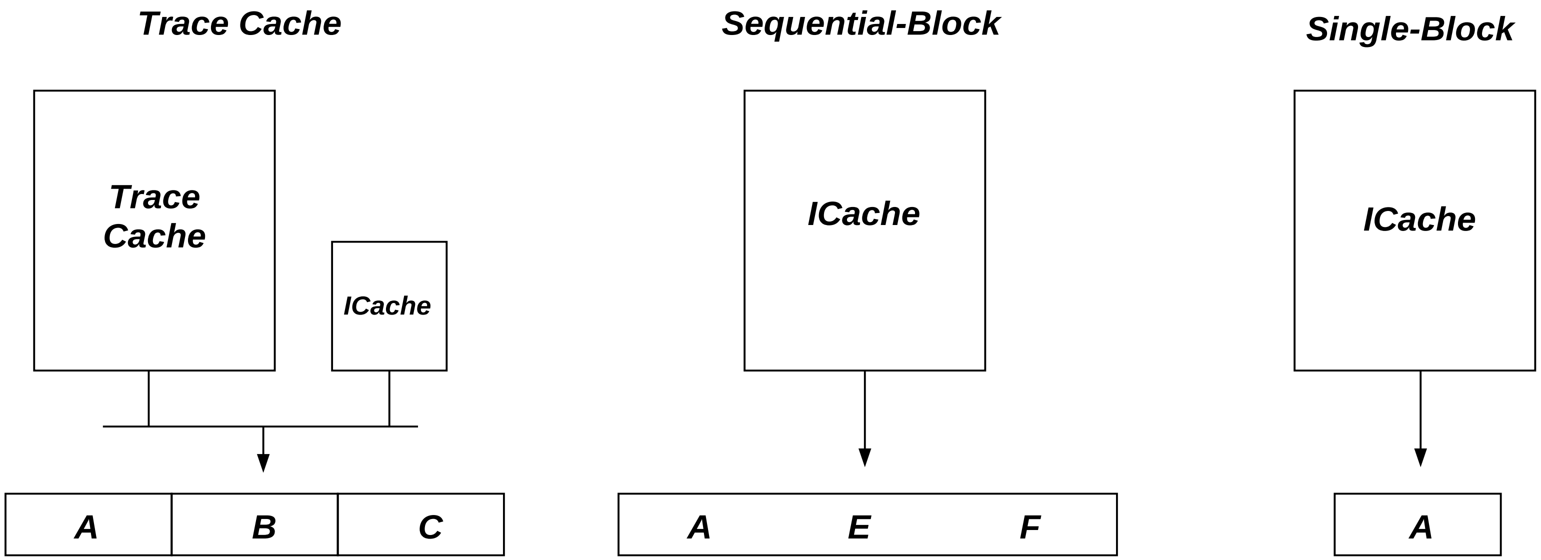
Large Execution Atomic Units : Melvin, Patt	1989
Intel Patent : Peleg, Weiser	1994
Trace Cache : Rotenberg et al., Patel et al.	1996-7

Performance

Experimental infrastructure

Simulator	Built upon SimpleScalar3.0 Simulates Alpha AXP code Models out-of-order execution (HPS)
Benchmarks	8 SPECint95 + 5 UNIX apps Aggressive compiler optimizations
Analysis Tools	Dependency graphs Trace segments Branches

Baseline configurations



Trace Cache

Sequential-Block

Single-Block

Baseline specifics

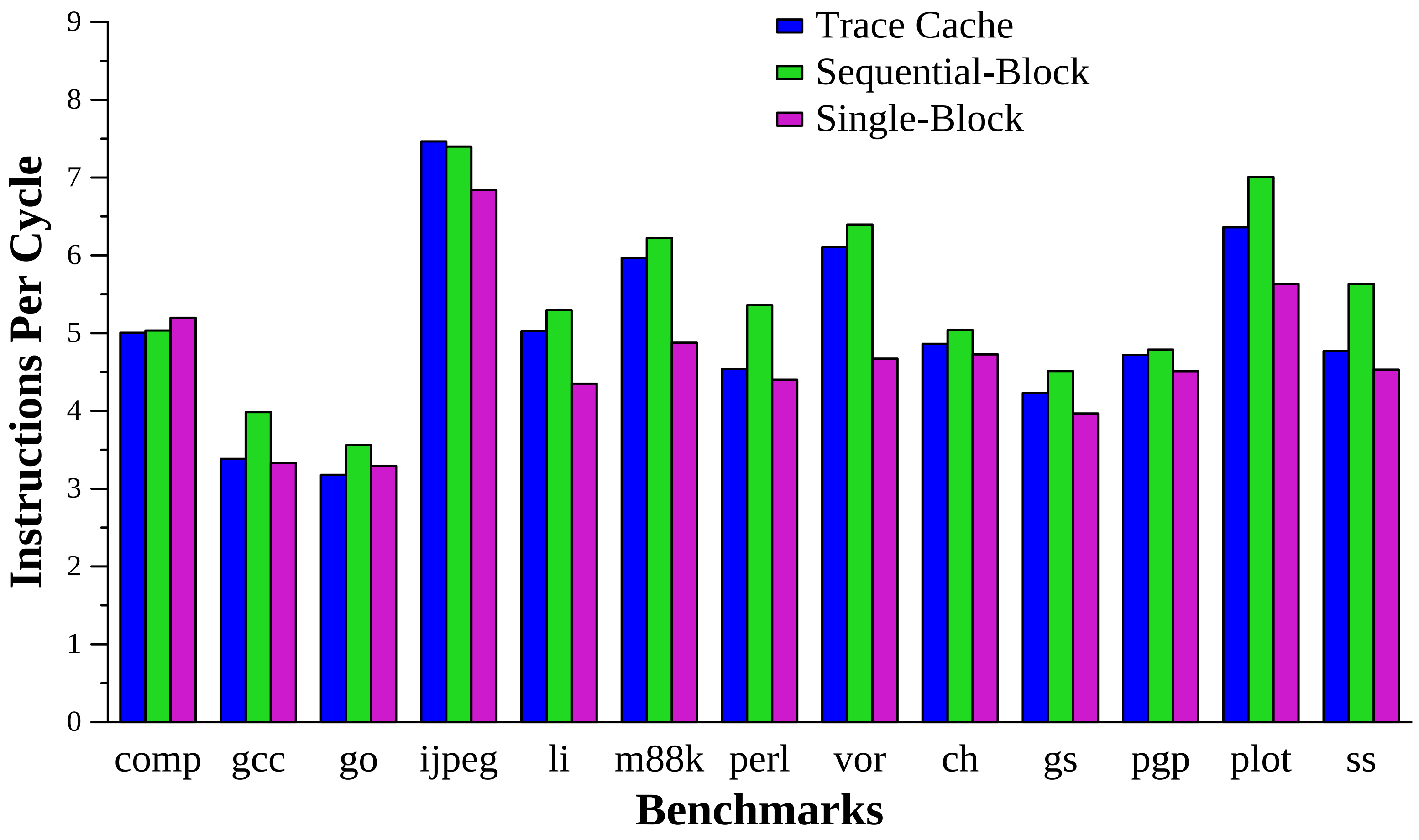
	Trace Cache	Sequential	Single
TCache size	128KB	—	—
ICache size	4KB	128KB	128KB
Blocks per fetch	3	3	1
Branch predictor	Multiple	Multiple	Hybrid
Execution units	16	16	16
DCache size	64KB	64KB	64KB
Inst window	512 insts	512 insts	512 insts
Mem dependence	Perfect	Perfect	Perfect

How well do the compiler optimizations do?

Instructions per taken branch

	No Code Layout	With Code Layout
compress	13.07	26.07
gcc	8.46	14.02
go	9.11	15.05
ijpeg	19.14	31.65
li	8.41	10.85
m88ksim	5.78	14.86
perl	9.13	10.04
vortex	11.05	12.45
Average	10.52	16.87

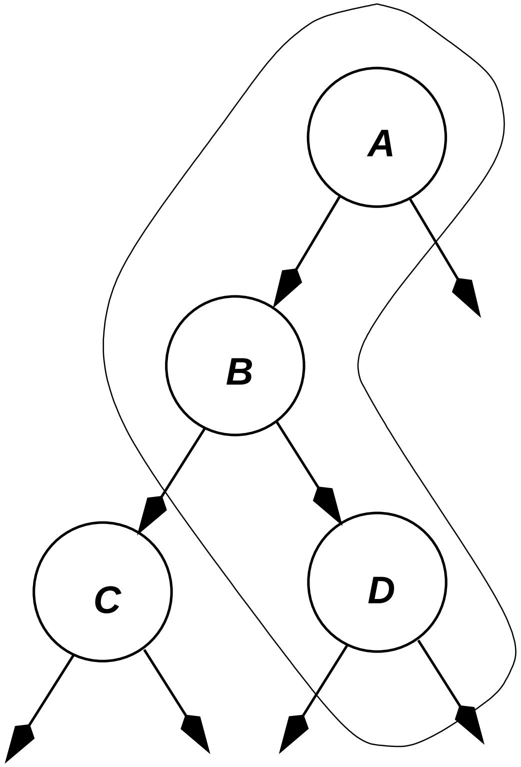
First measurement



Average fetch misses per 1000 instructions

Trace Cache	Sequential-Block	Single-Block
33.69	0.19	0.19

Partial Matching

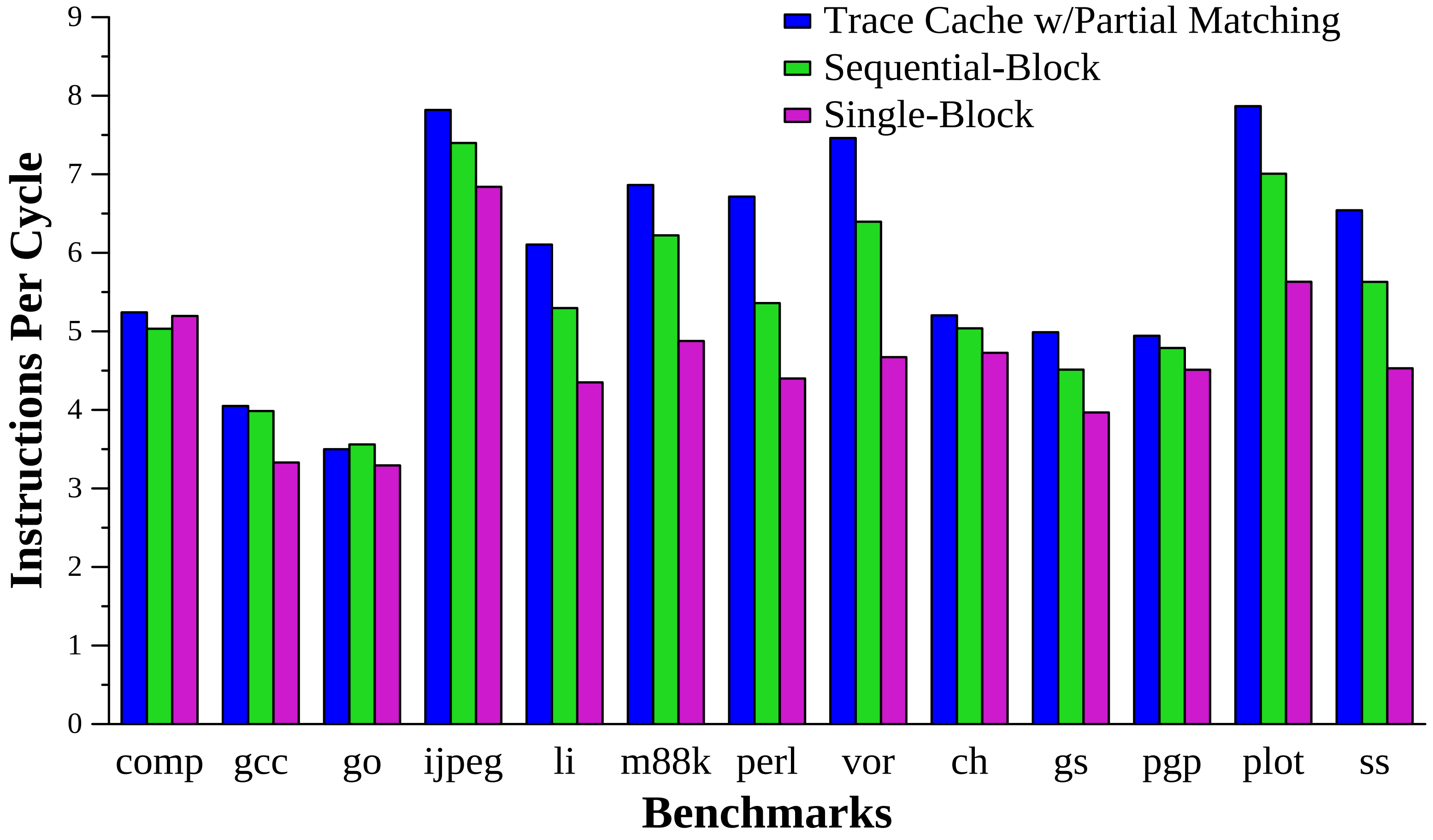


Predicted path: ABC
Fetched segment: ABD

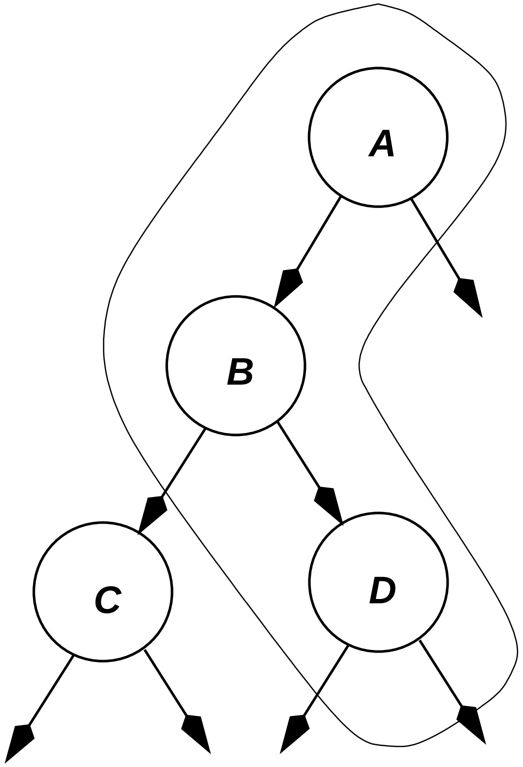
No Partial Matching: miss

Partial Matching: AB

Partial Matching results



Inactive Issue

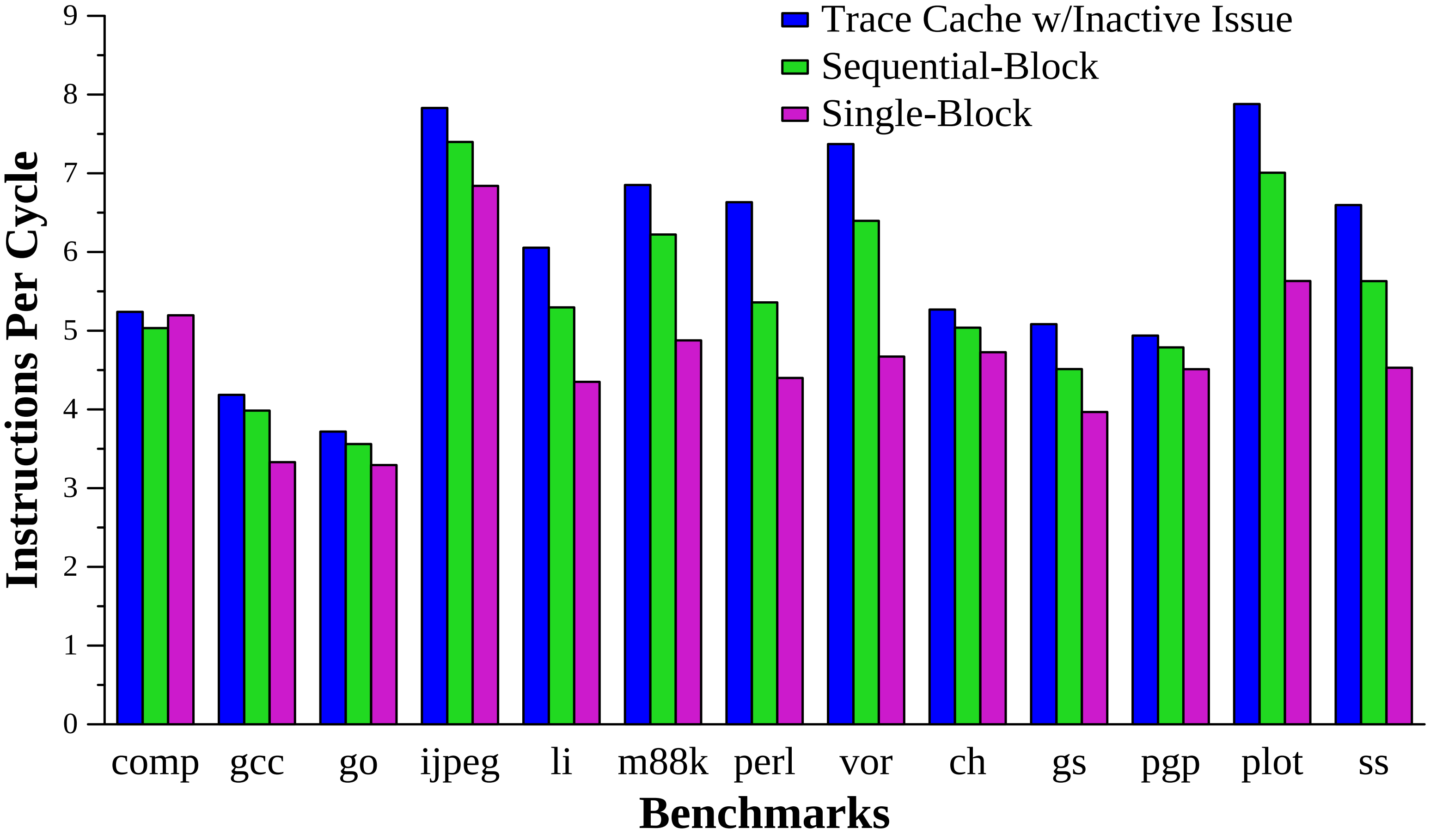


Predicted path: ABC
Fetched segment: ABD

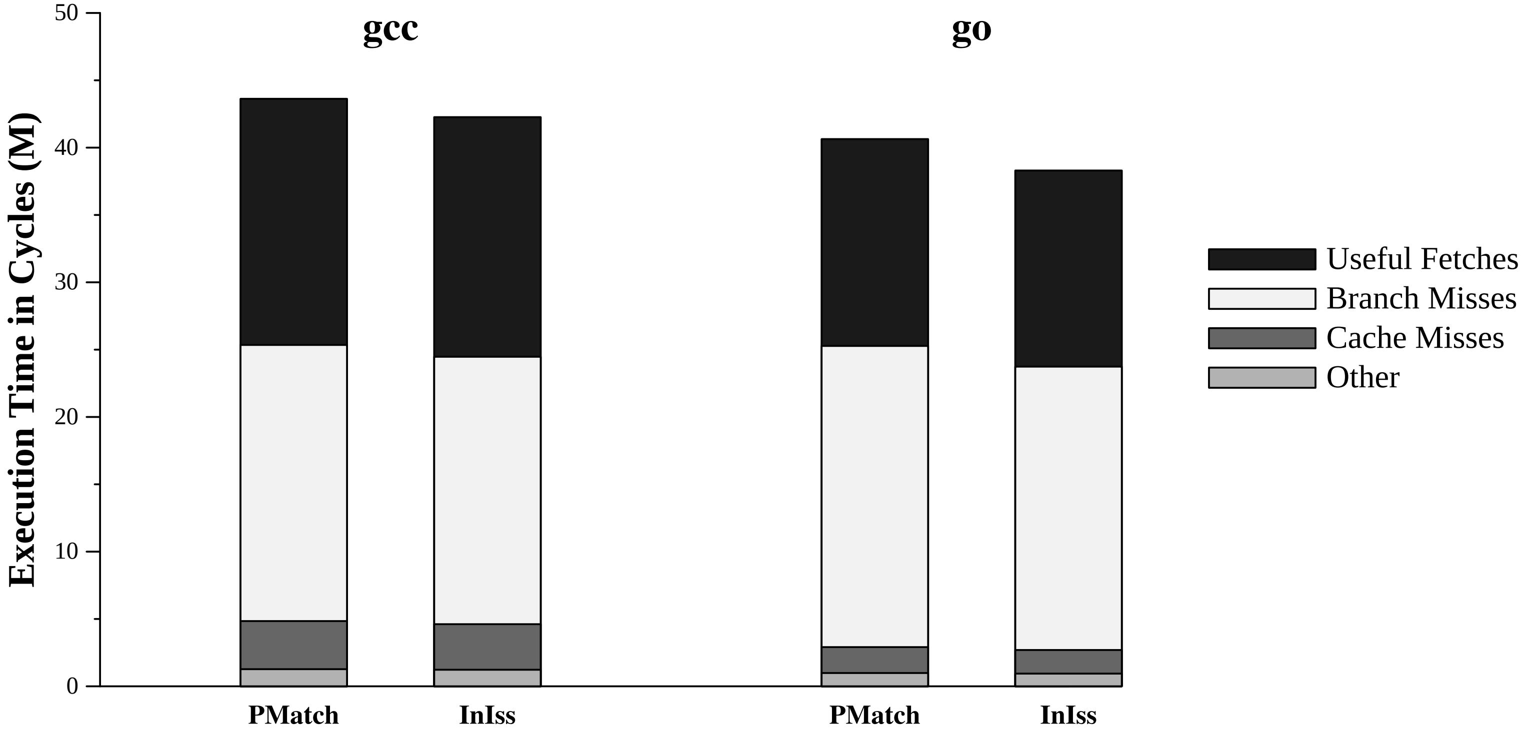
No Partial Matching: miss
Partial Matching: AB

Inactive Issue: AB (active) D(inactive)

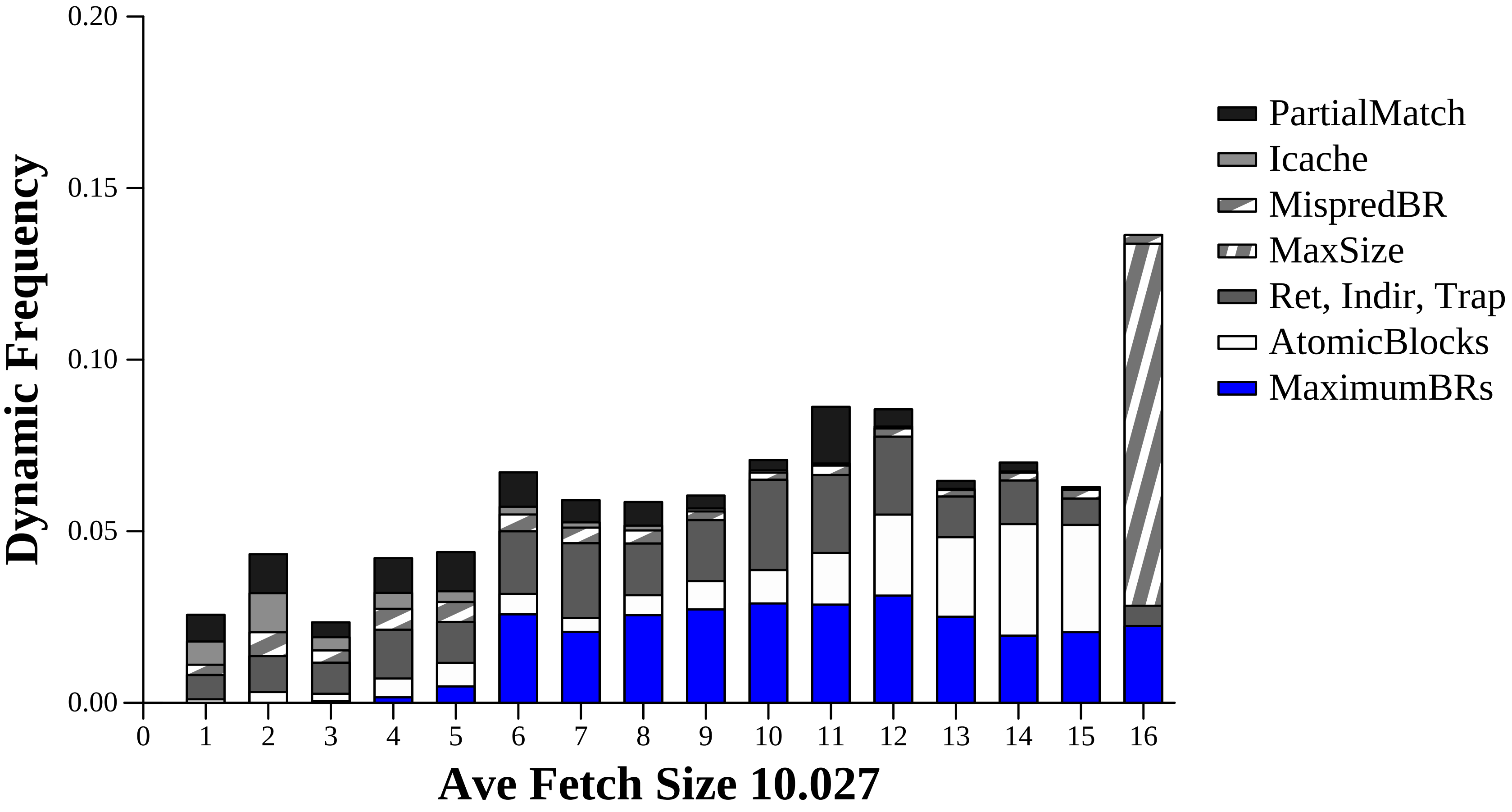
Inactive Issue results



Inactive Issue reduces branch penalty

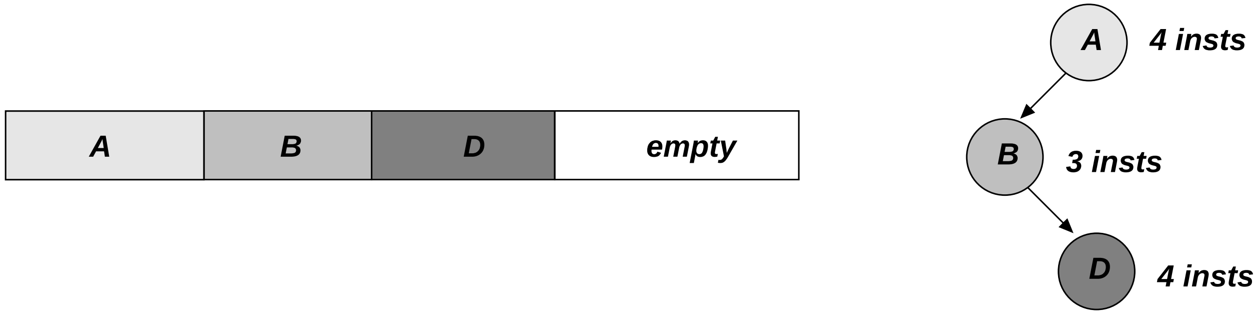


Limitations on fetch size: gcc

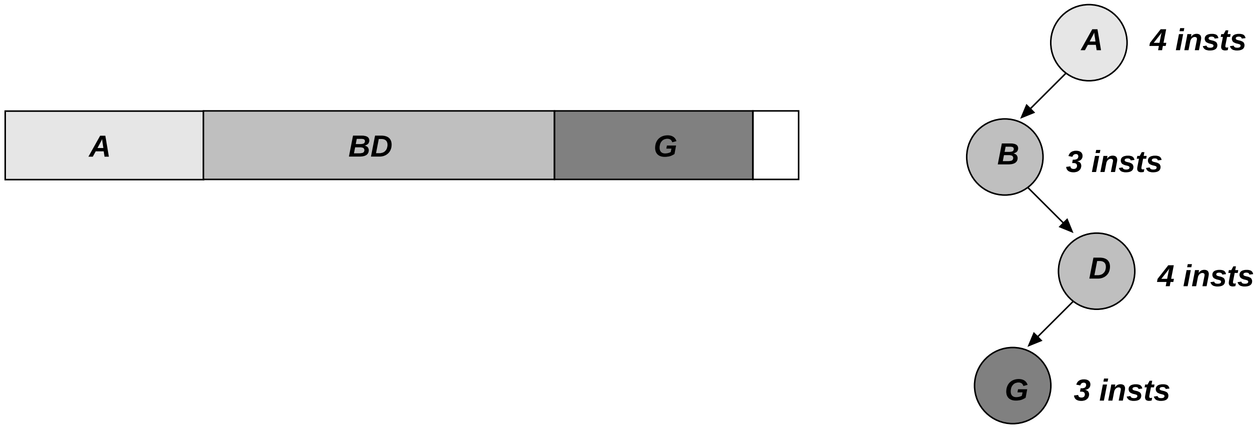


Branch Promotion

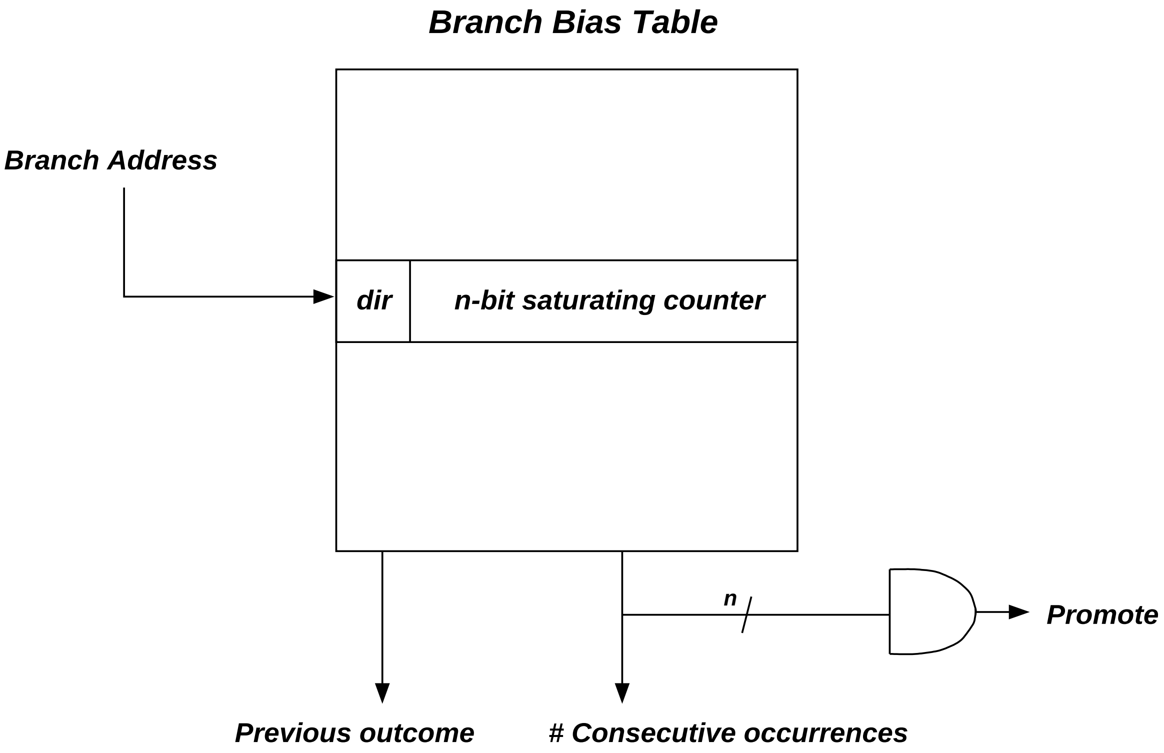
Without Branch Promotion



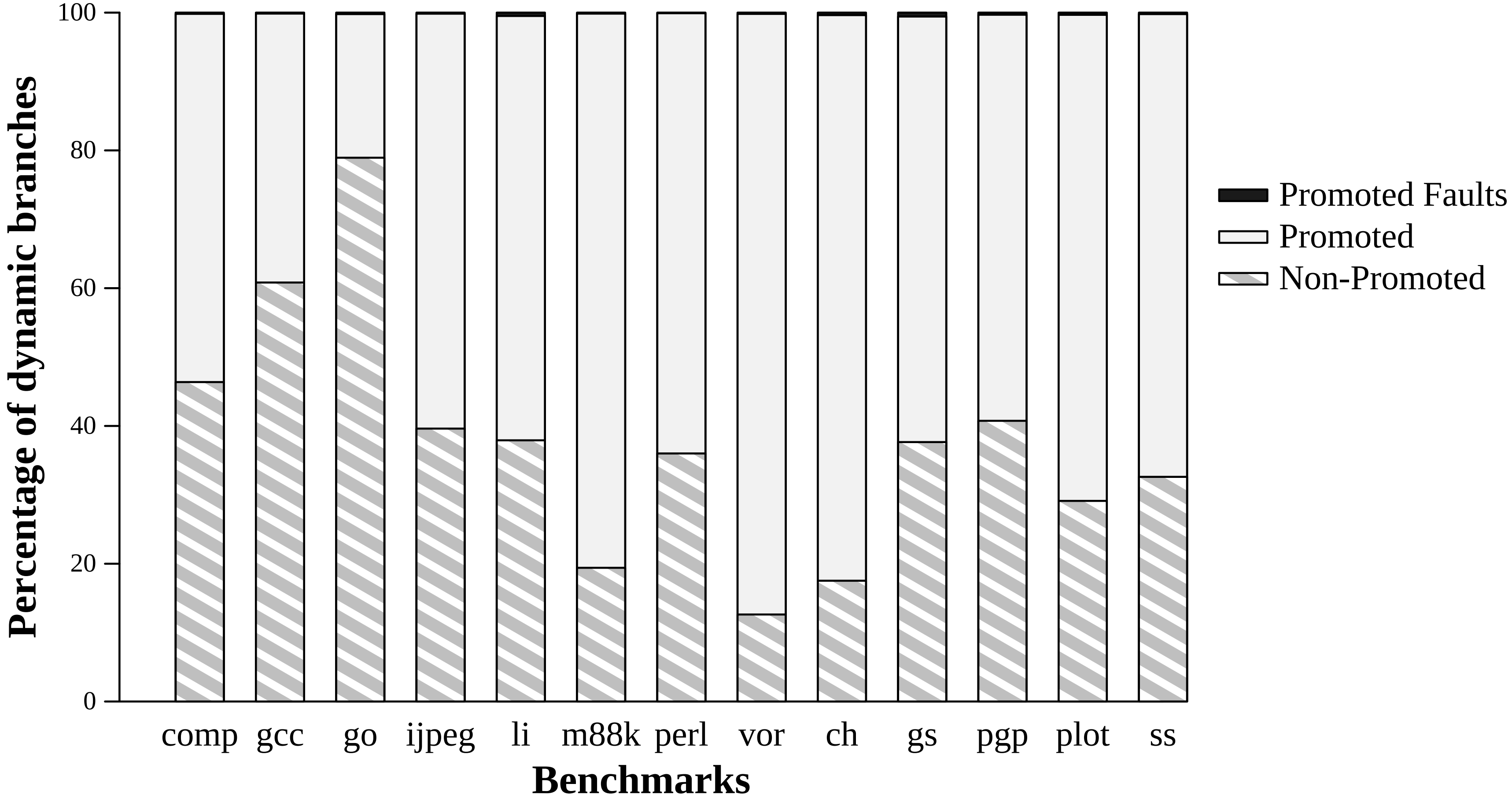
With Branch Promotion



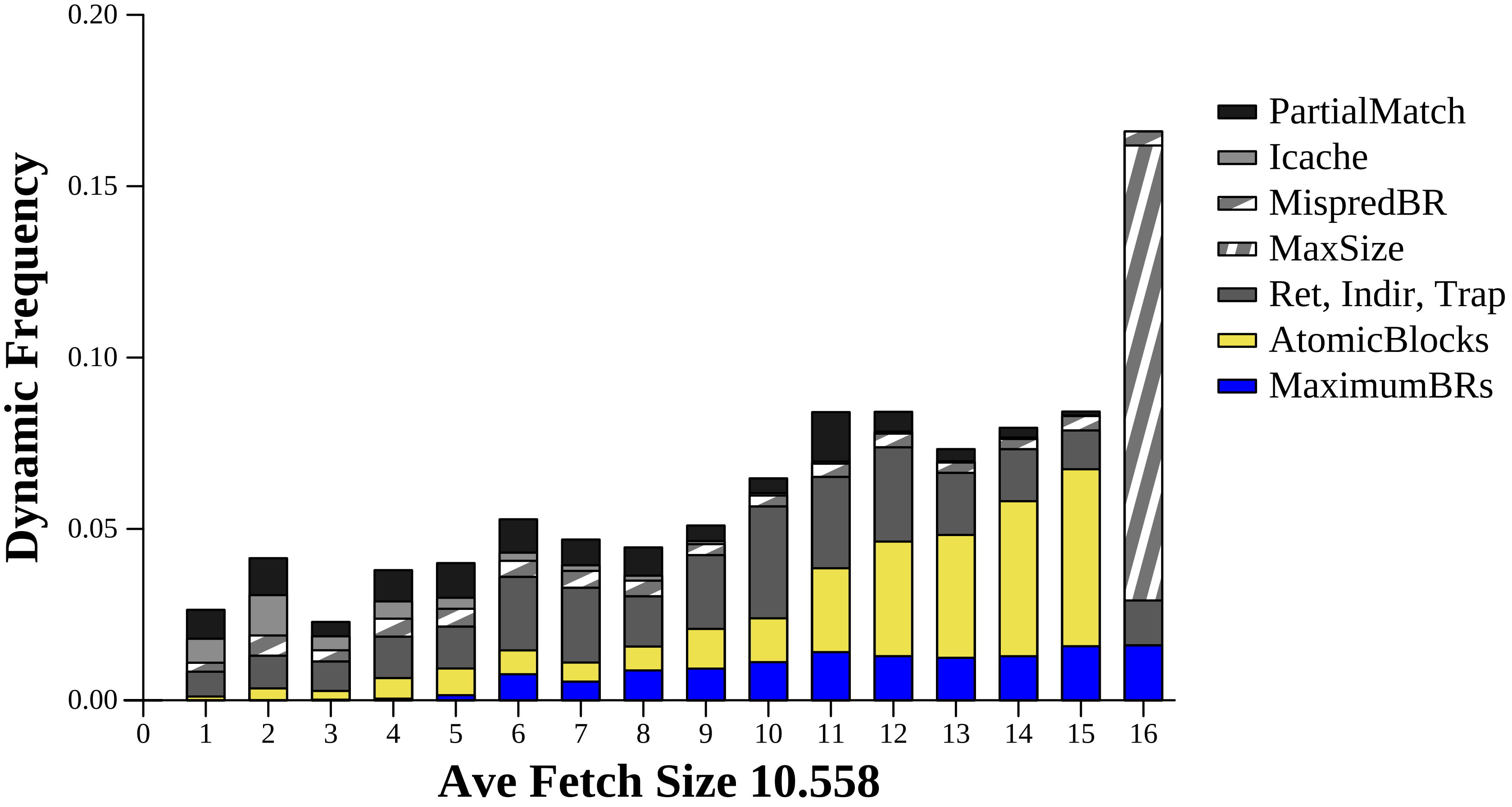
Implementing Branch Promotion



Promoted branch frequency

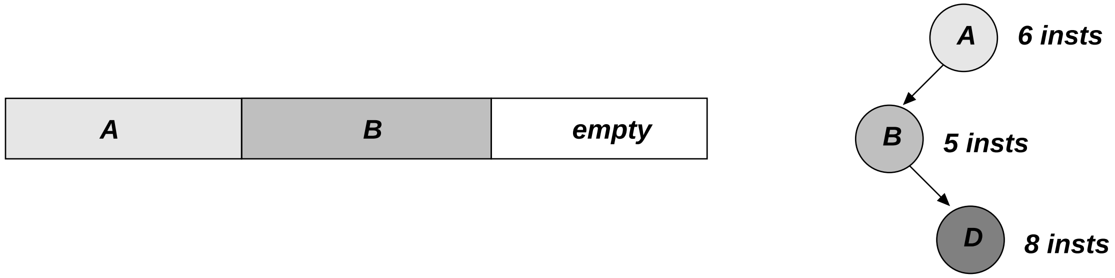


Limitations on trace size: gcc with Promotion

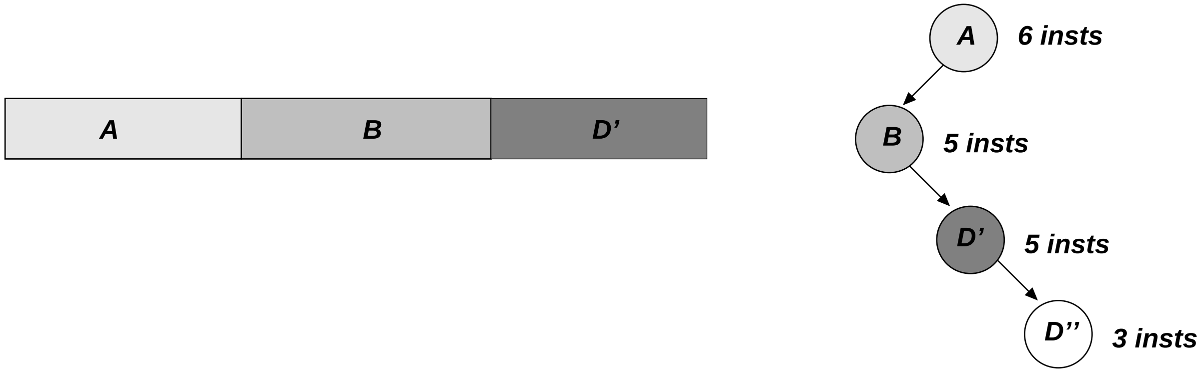


Trace Packing

Without Trace Packing



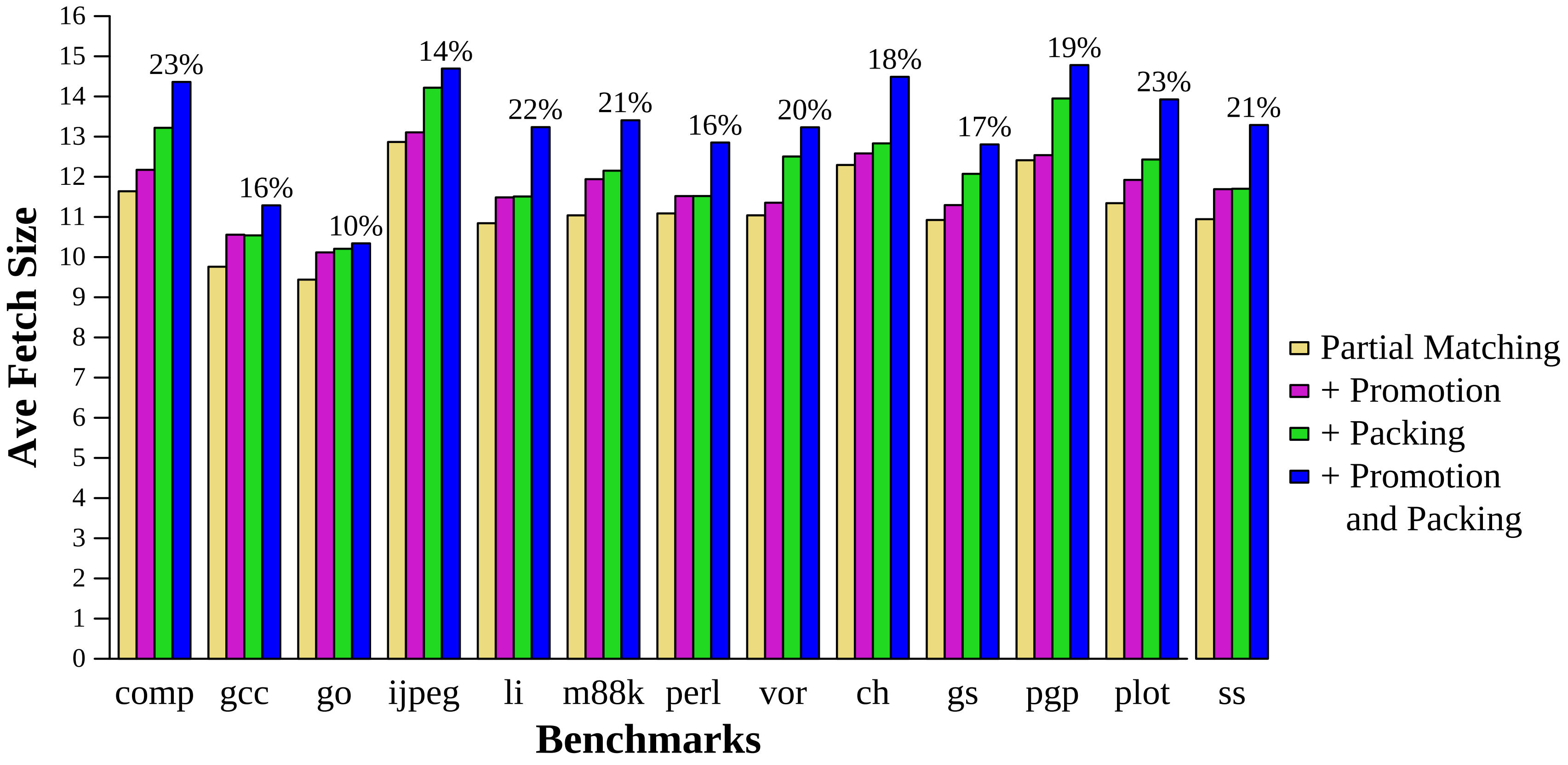
With Trace Packing



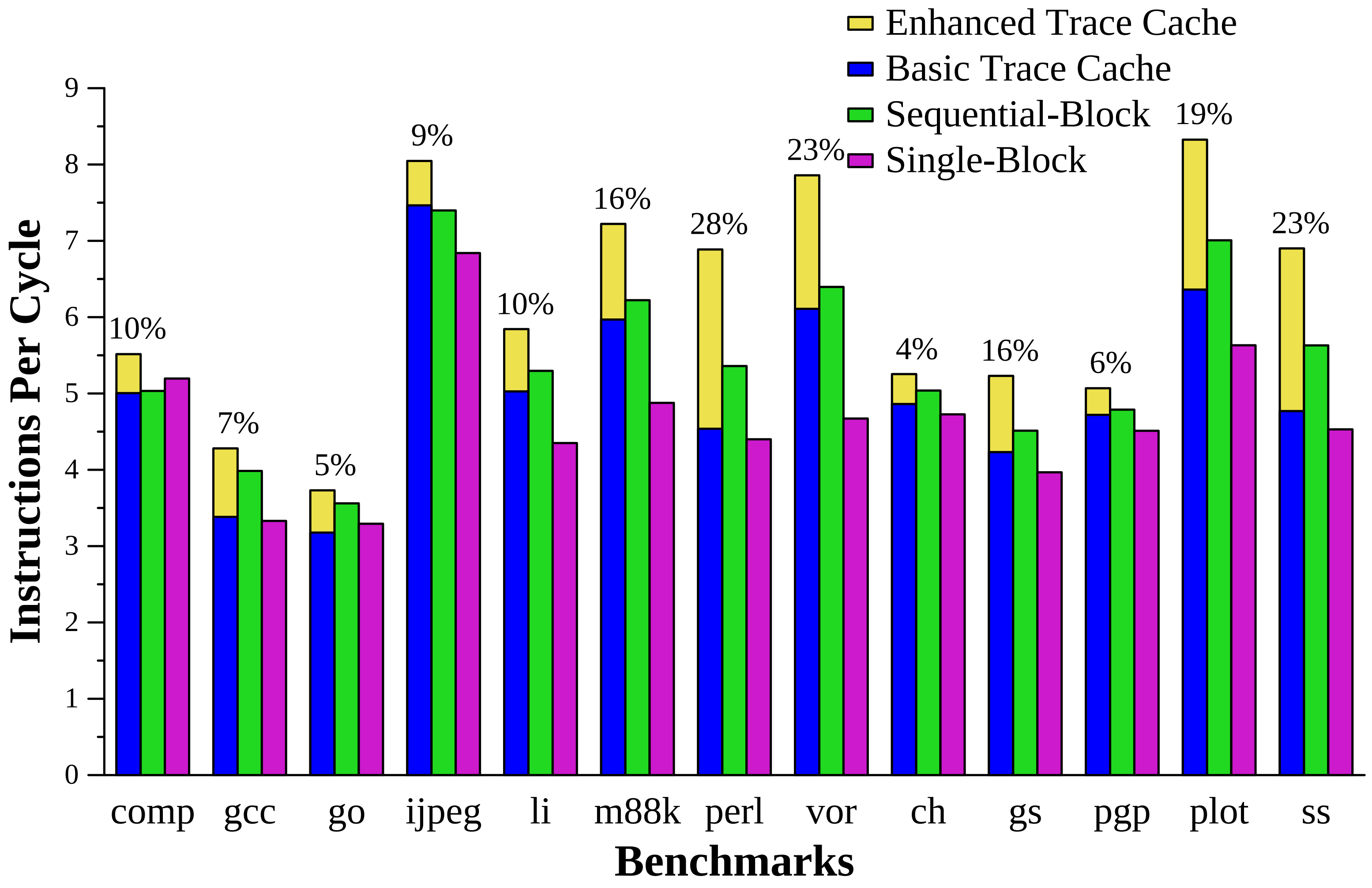
Three Packing schemes

- Unregulated
- n-Regulated
  - only pack chunks of n instructions
- Cost-Regulated
  - only pack if unused space is large OR line contains backwards branch

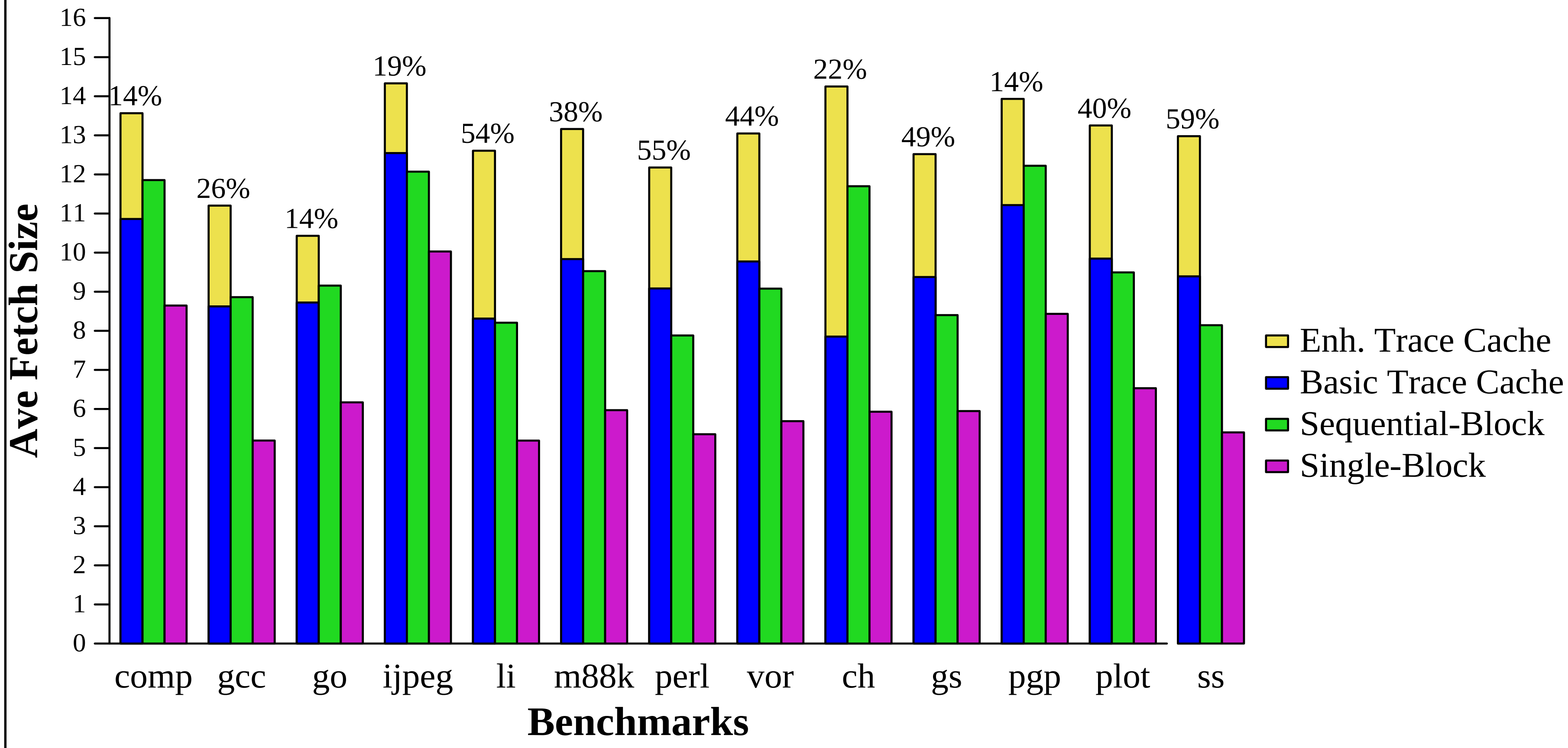
Average fetch size



Performance w/enhancements



Average fetch size w/enhancements

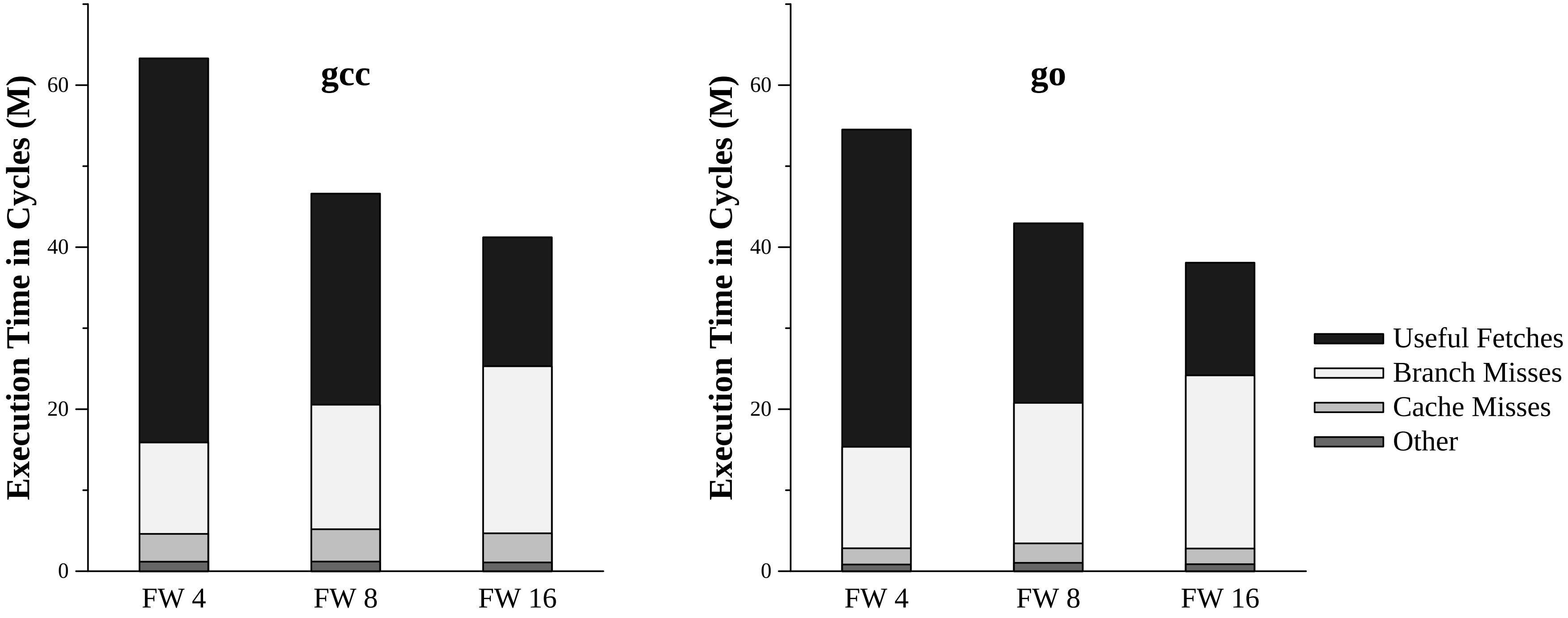


34% increase in fetch rate.

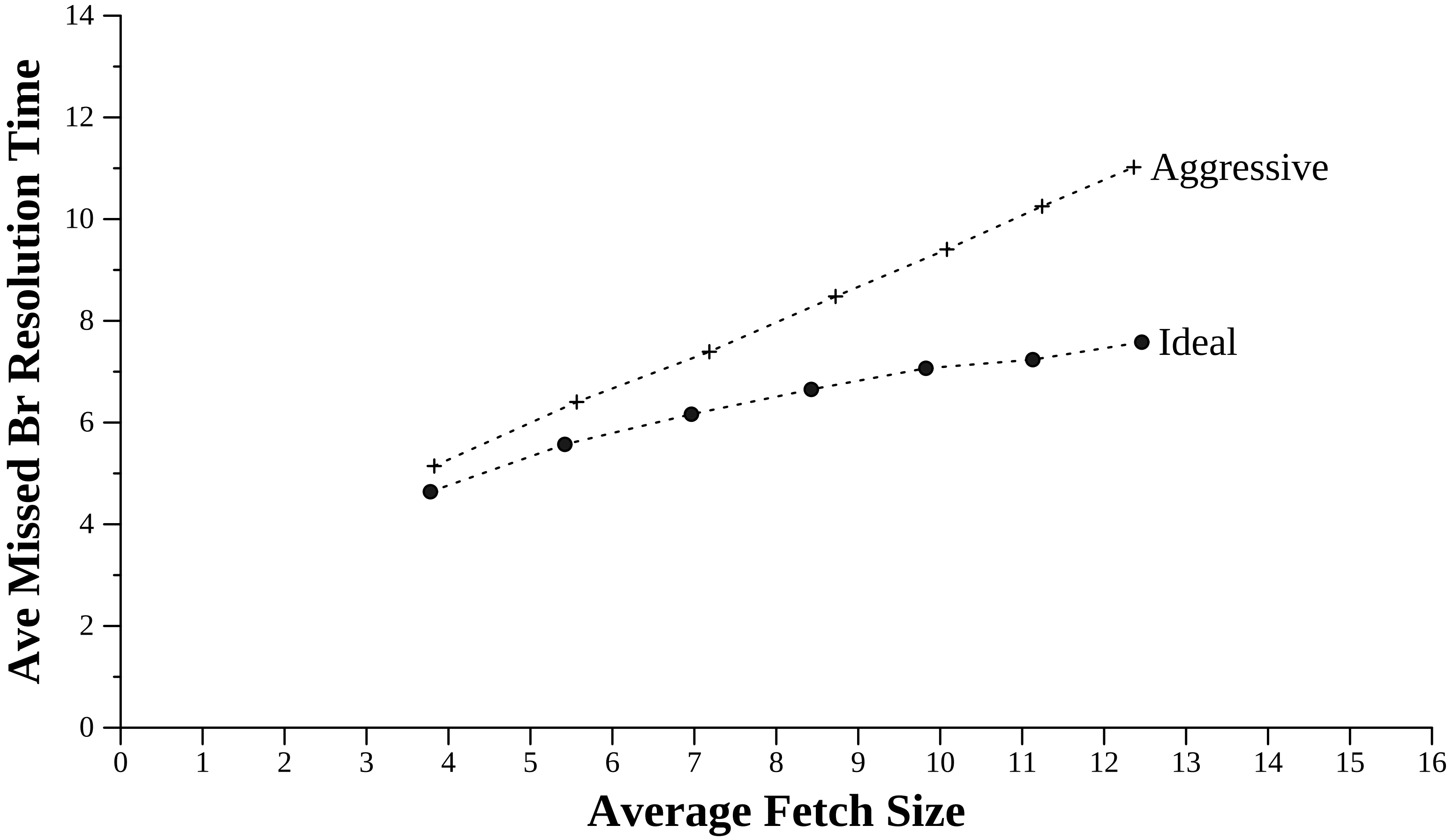
But only a 14% increase in performance.

Why?

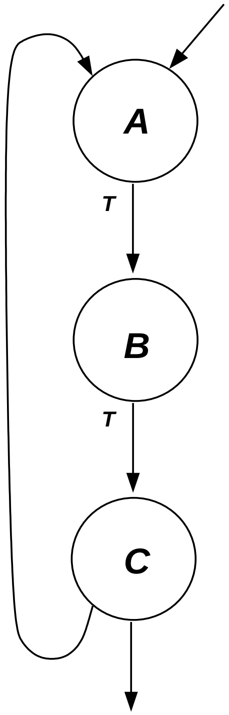
Effect on branch resolution time



Branch resolution time vs. fetch rate



Duplication in the trace cache



6 instructions

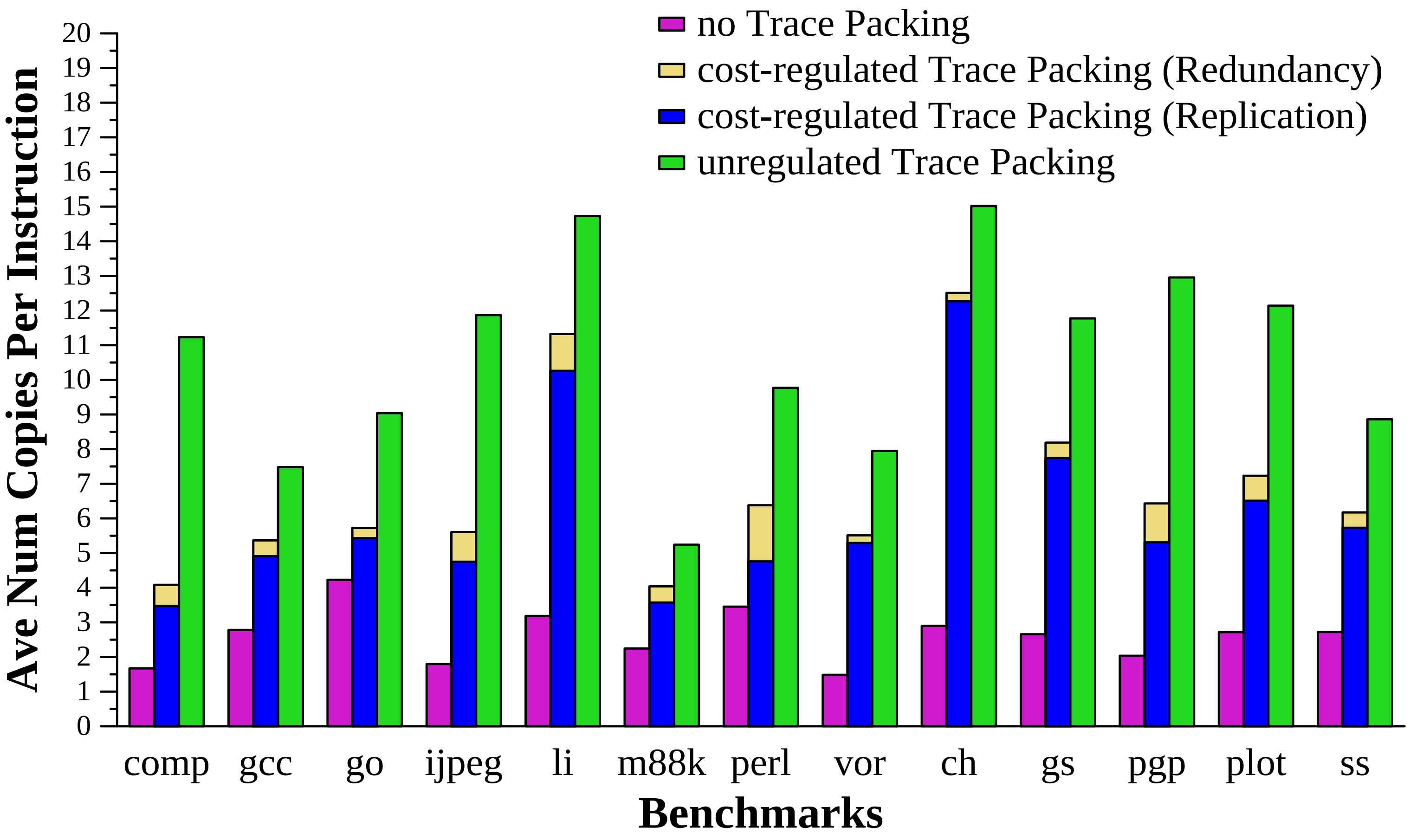
5 instructions

6 instructions

ICache: No duplications
1 iteration every 3 cycles

Enhanced Trace Cache: 6 segments
2.5 iterations every 3 cycles

Measure of duplication

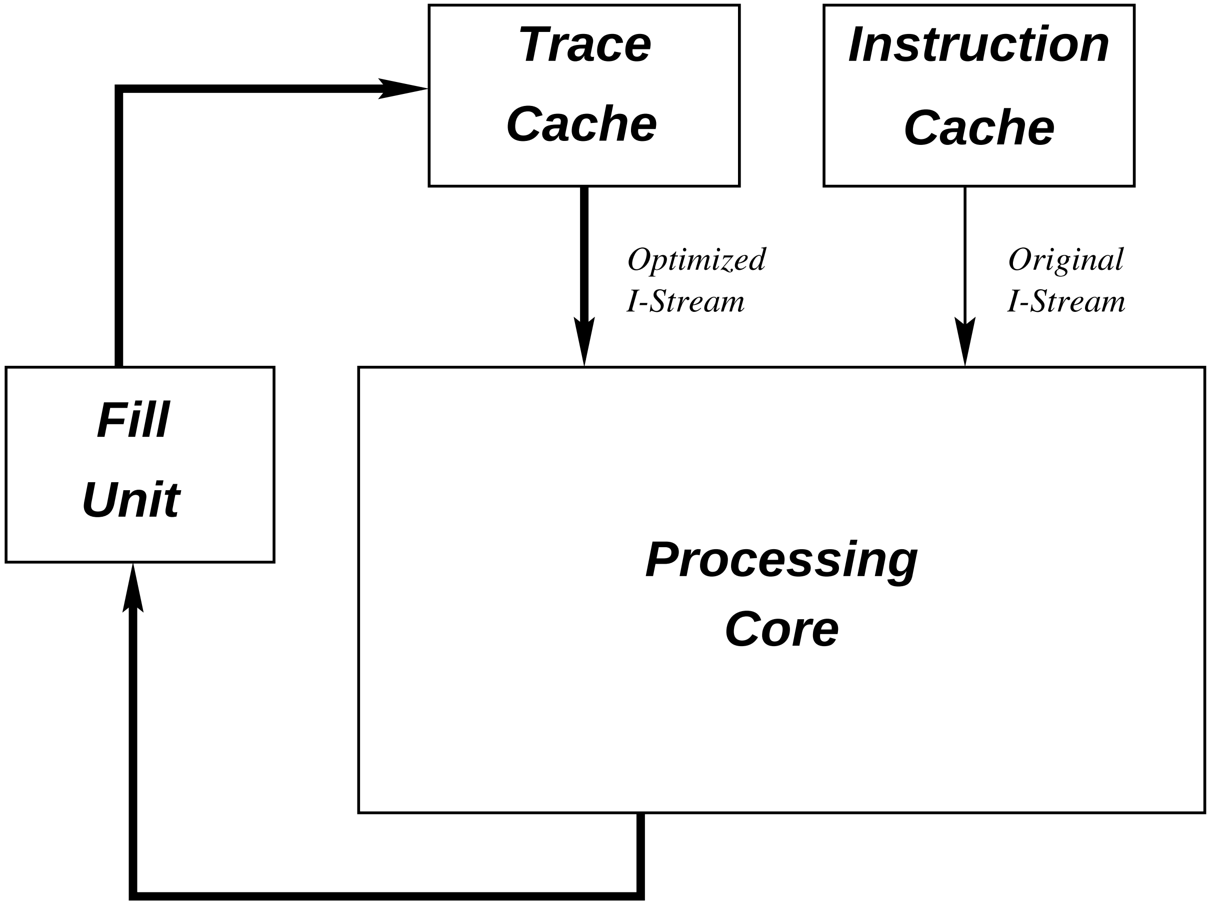


Finally...

Major Findings

- Enhanced TCache outperforms Sequential ICache
  - IPC increases by 14%
  - Fetch rate increases by 34%
- 60% fewer dynamic cond. branches with Promotion
- Mispredictions more costly at higher fetch rates
- Duplication high, but most copies useful

Future Directions:
Dynamic Trace Optimizations



Future Research Plans

- Short term
  - Enhancing the Sequential-Block ICache
  - Fill unit optimizations
  - Compiler-assisted trace segment creation
- Longer term
  - Fundamental analysis of programs
  - General-purpose to application-specific
  - Programming for high performance