Trace Processors

Eric Rotenberg

Quinn Jacobson, Yanos Sazeides, Jim Smith

Computer Sciences Department
University of Wisconsin — Madison
http://www.cs.wisc.edu/~ericro/ericro.html

Introduction

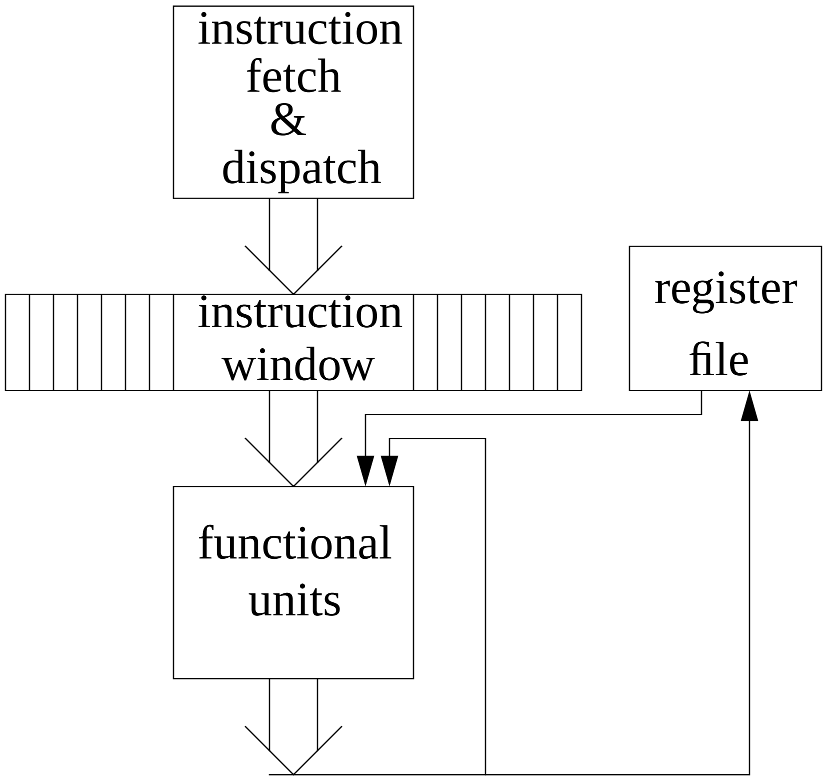
- Goal: issue many instructions per cycle, *and* keep cycle times fast
- What we have now: dynamically scheduled, modest superscalar processors
- Problem: is conventional superscalar a good candidate for very wide-issue machines?
  - Complexity issues

    i.e. cycle time related

    efficiently exploiting instruction-level parallelism
  - Architectural issues

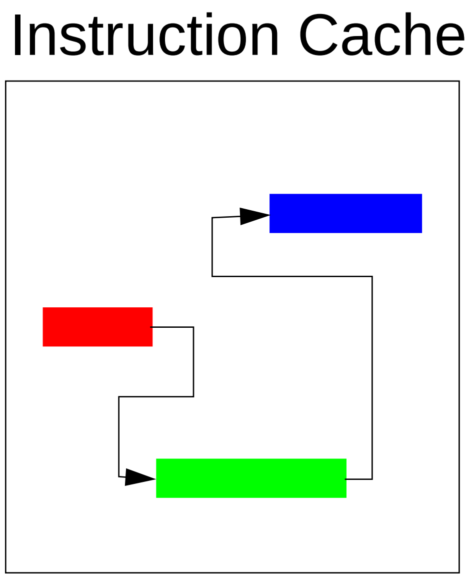
    exposing instruction-level parallelism

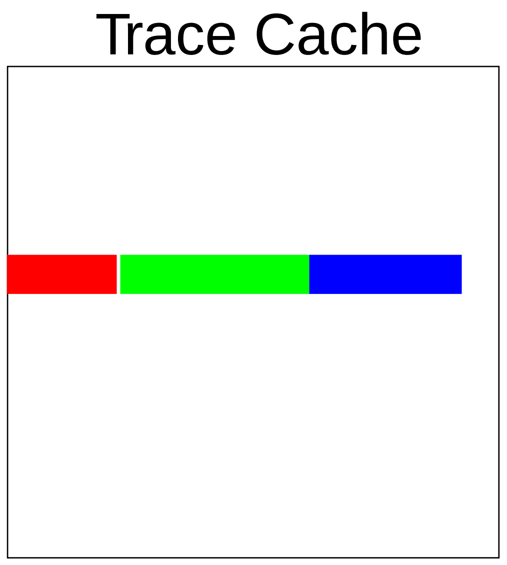
Superscalar Organization



What is a Trace?

- A *trace* is a dynamic sequence of instructions captured and stored by hardware
  - Traces are built as the program executes
  - Stored in a trace cache





Trace property 1: control hierarchy

- **A trace can contain any number and type of control transfer instructions, i.e. any number of implicit control predictions**
  - Unit of control prediction should be a trace, not individual branches
  - Suggests a next-trace predictor

Trace property 2: data hierarchy

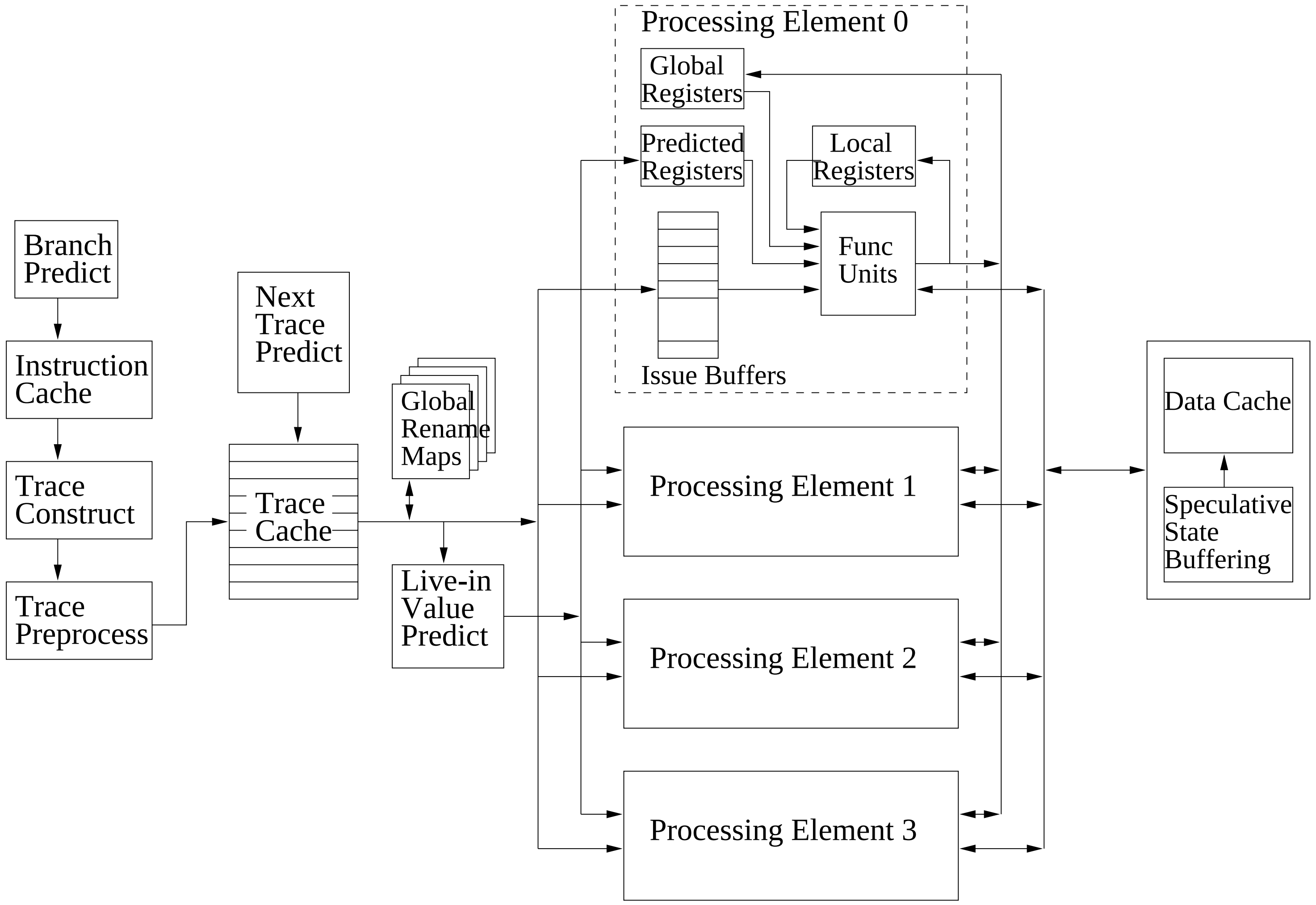
- **A trace uses and produces values that are either live-on-entry, entirely local, or live-on-exit**

  terms: ***live-ins***, ***locals***, and ***live-outs*** respectively

  - Suggests a hierarchical register file: a local register file per trace for local values, a single global file for values live between traces. Pre-rename local values.
  - Local (intra-trace) dependences and global (inter-trace) dependences suggest distributing instruction window based on trace boundaries

[Vajapeyam,Mitra] and [Franklin,Sohi]

Trace Processor



Hierarchy: overcoming complexity

- Instruction fetch: trace cache and next-trace predictor take care of instruction fetch bottleneck
- Instruction dispatch: only global values are renamed, and no dependence checking
- Instruction issue: distributed wakeup and select logic
- Result bypassing: full bypassing within a PE, delayed bypassing between PEs
- Register file: global register file can be smaller, fewer ports
- Instruction retirement: the dual of dispatch

Speculation: exposing ILP

- Control dependences
  - next-trace prediction can yield better overall branch prediction accuracy than many aggressive single-branch predictors
- Data dependences
  - value prediction and speculation
  - structured value prediction: predict only live-ins
- Memory dependences
  - predict all load and store addresses
  - loads issue speculatively as if no prior stores

Handling misspeculation

1. An instruction reissues when it detects any type of mispredict: value, address, memory dependence, and control (register dependence)
   - Paper proposes a collection of mechanisms for detecting all kinds of mispredictions
2. Selective reissuing of dependent instructions
   - Occurs naturally via the existing issue mechanism, i.e. the receipt of new values, and is independent of the mispredict origin

End result: a dynamic instruction can issue any number of times between dispatch and retirement.

Related Work

- Multiscalar processors - Franklin, Vijaykumar, Breach, Sohi
- Trace window organization - Vajapeyam, Mitra
- Dependence-based clustering - Palacharla, Jouppi, Smith
- Fill unit - Melvin, Shebanow, Patt
- Data prediction - Lipasti,Shen / Sazeides,Smith

Companion work:

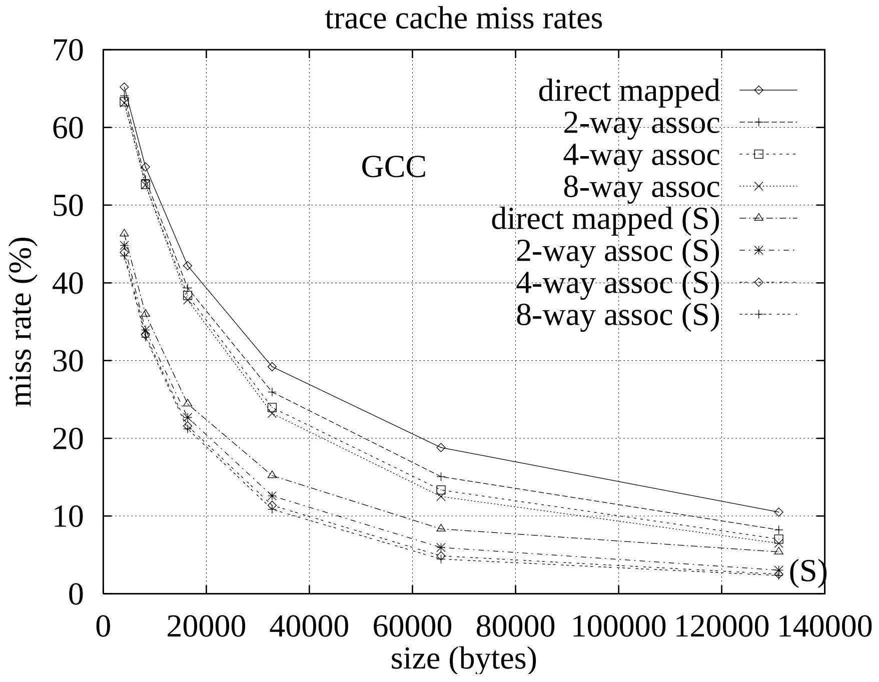
- Context-based value prediction - Sazeides, Smith
- Next-trace prediction - Jacobson, Rotenberg, Smith
- Trace cache - Peleg,Weiser / Rotenberg,Bennett,Smith / Patel,Friendly,Patt

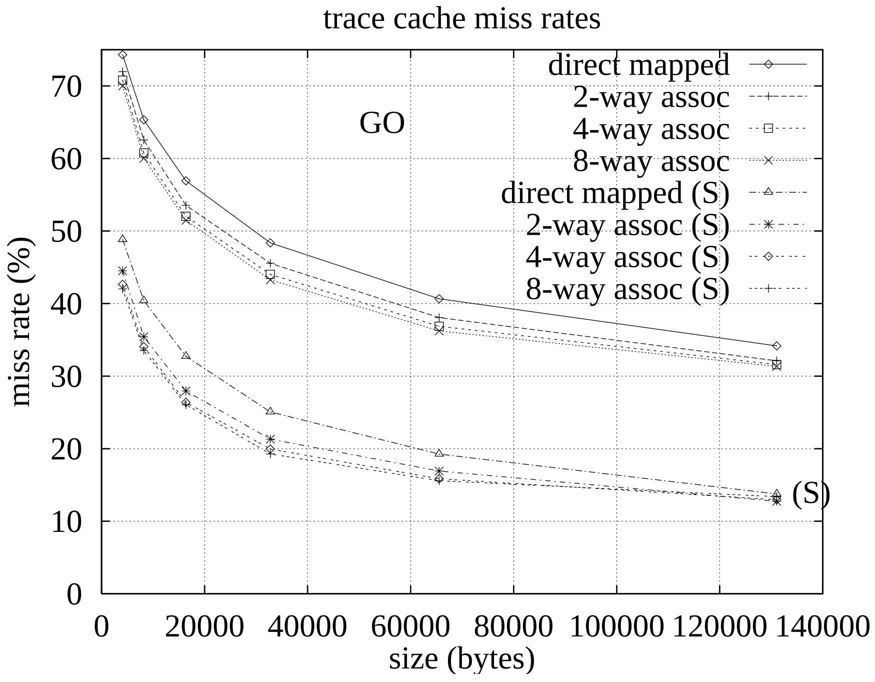
Trace Selection

- Trace selection
  - algorithm used to delineate traces
  - interesting tradeoffs to optimize for: trace length, PE utilization and load balance, trace cache hit rate, trace prediction accuracy, control independence, ...
- Some heuristics
  - stop at or embed various types of control instructions
  - stop at loop edges, ensure stopping at basic block boundaries, remember past start-points
  - don’t stop at call direct if it’s a unique call site, embed leaf functions
  - reconvergent control flow
- Default trace selection
  - stop at a maximum of 16 instructions, or
  - stop at any call indirect, jump indirect, return

Trace Cache Performance

- *compress*: fits entirely in 16KB direct mapped trace cache
- *jpeg*, *xlisp*: 4% miss rates for 32KB direct mapped trace cache
- avg trace lengths: *gcc* {13.9, 10.9}, *go* {14.8, 11.8}





Next-Trace and Value Predictors

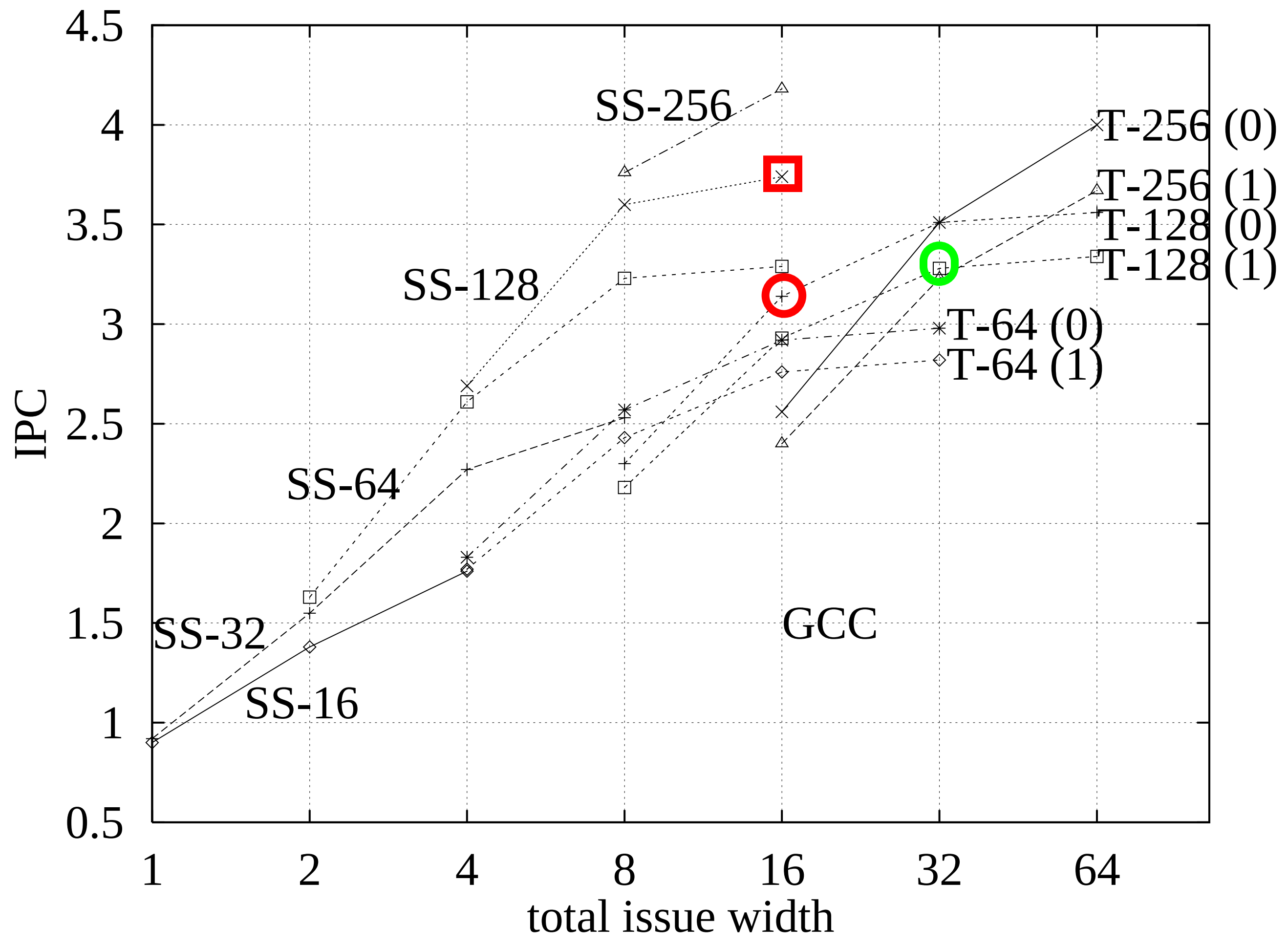
- Trace prediction
  - correlated predictor that uses the path history of previous traces
  - outputs next trace and one alternate prediction for fast recovery
  - *Hear Quinn’s talk*
- Value prediction
  - context-based: learns values that follow a particular sequence of previous values
  - outputs 32-bit value and indicates confident or not
  - *Hear Yanos’s talk*

Experiments

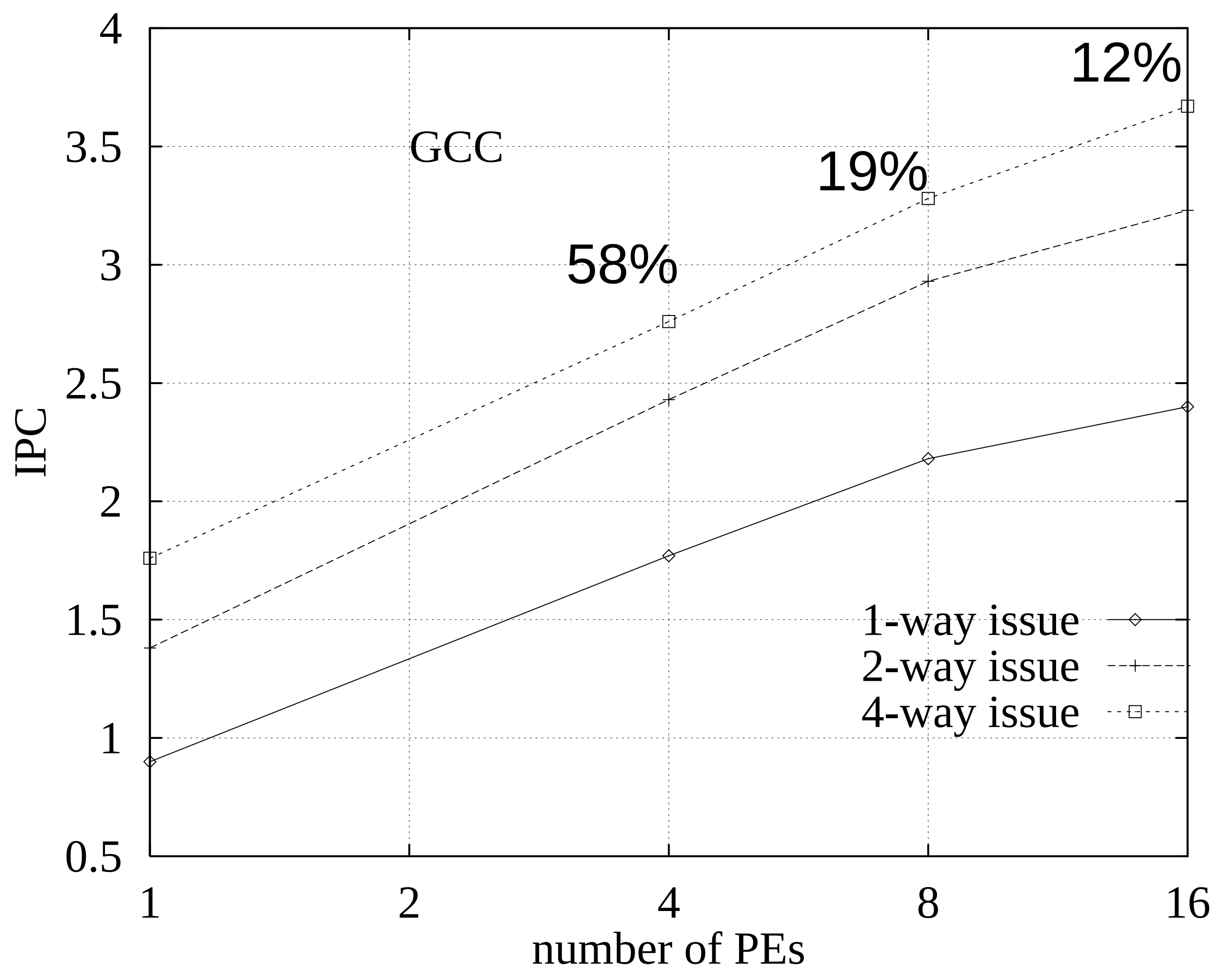
- Three sets of experiments:

  1. Primary performance results: both superscalar and trace processors, no value prediction and uses conventional control flow model
  2. A trace processor with structured value prediction
  3. A trace processor with aggressive control flow model

gcc


IPC
total issue width
4.5
4
3.5
3
2.5
2
1.5
1
0.5
1
2
4
8
16
32
64
SS-256
SS-128
SS-64
SS-32
SS-16
T-256 (0)
T-256 (1)
T-128 (0)
T-128 (1)
T-64 (0)
T-64 (1)
GCC

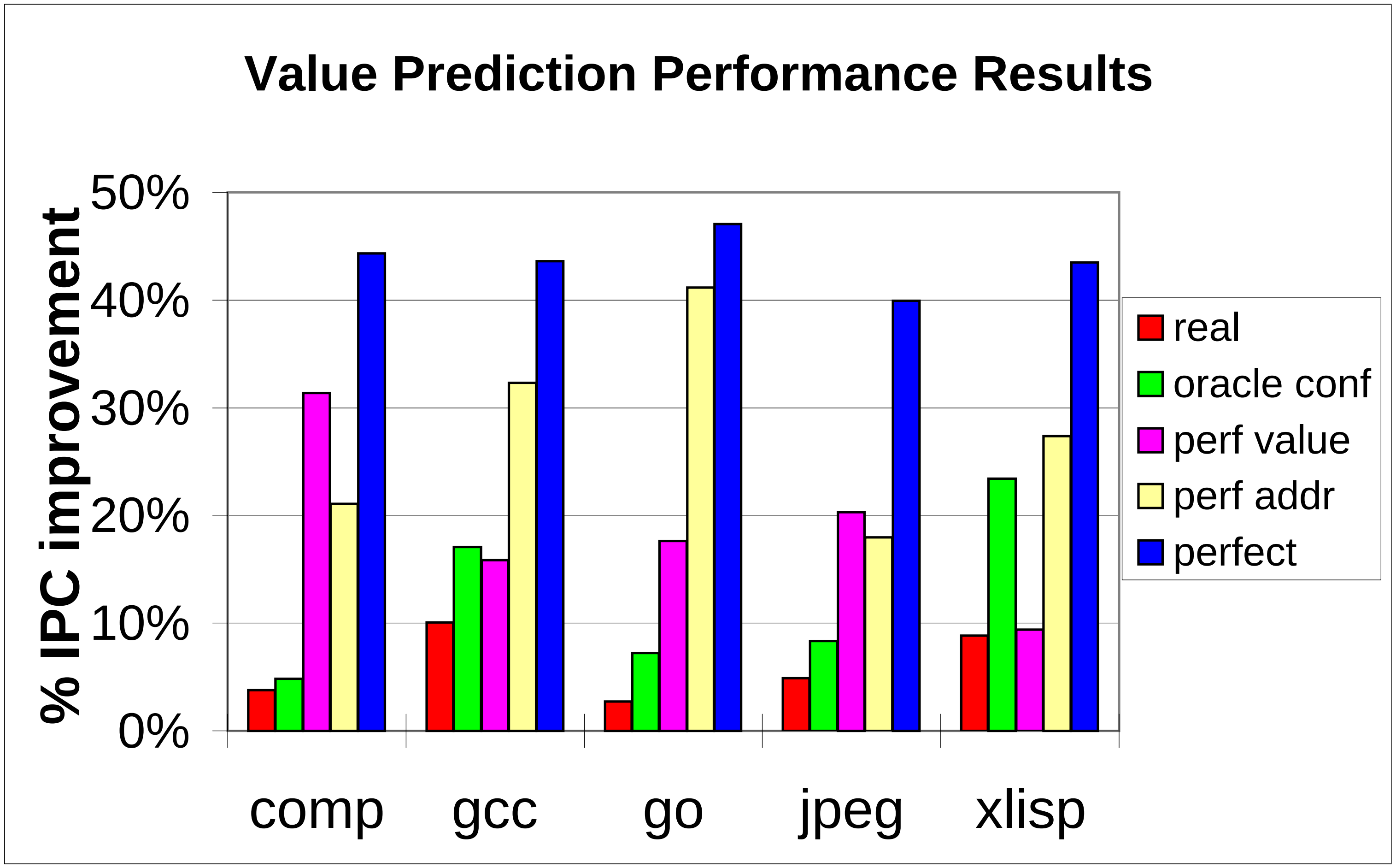
gcc



GCC
58%
19%
12%
IPC
number of PEs
1-way issue
2-way issue
4-way issue
0.5
1
1.5
2
2.5
3
3.5
4
1
2
4
8
16

Adding Value Prediction

- single configuration: “T-128 (1) 4-way issue per PE”



Aggressive Control Flow

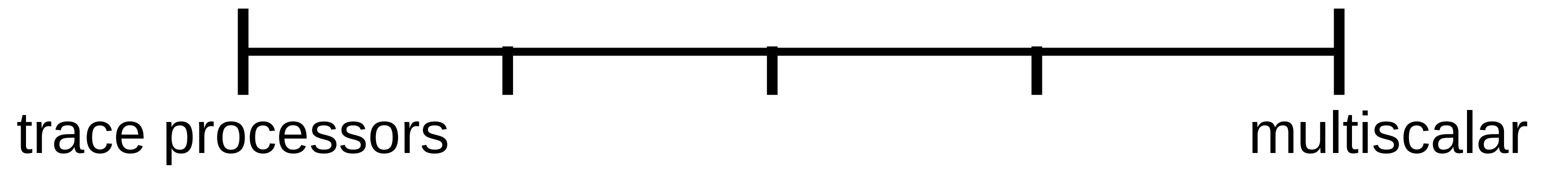
- With selective control mispredict recovery:
  - compress: 13% IPC improvement
  - jpeg: 9% IPC improvement
- Where is the benefit coming from?
  - frequent, small loops with simple reconvergent control flow
  - loops with few and fixed number of iterations
- Trace selection and more flexible PE allocation can improve exposure of control independence

Summary

- Trace processors exploit characteristics of traces
  - Control hierarchy: trace is unit of control prediction
  - Data hierarchy: trace is unit of work
- Value prediction applied to inter-trace dependences
  - potential performance is significant
  - value prediction is in its infancy, needs work
- Interesting misspeculation model
  - selective reissuing is natural
  - attempt to treat all types uniformly
- Aggressive control flow model shows potential

Future Work

- Trace selection
  - trace length
  - trace prediction accuracy
  - trace cache performance
  - enhance control independence
  - overall live-in prediction accuracy
- Compare with multiscalar
  - identify key differences (tasks vs. traces)
  - quantify advantages/disadvantages


trace processors
multiscalar