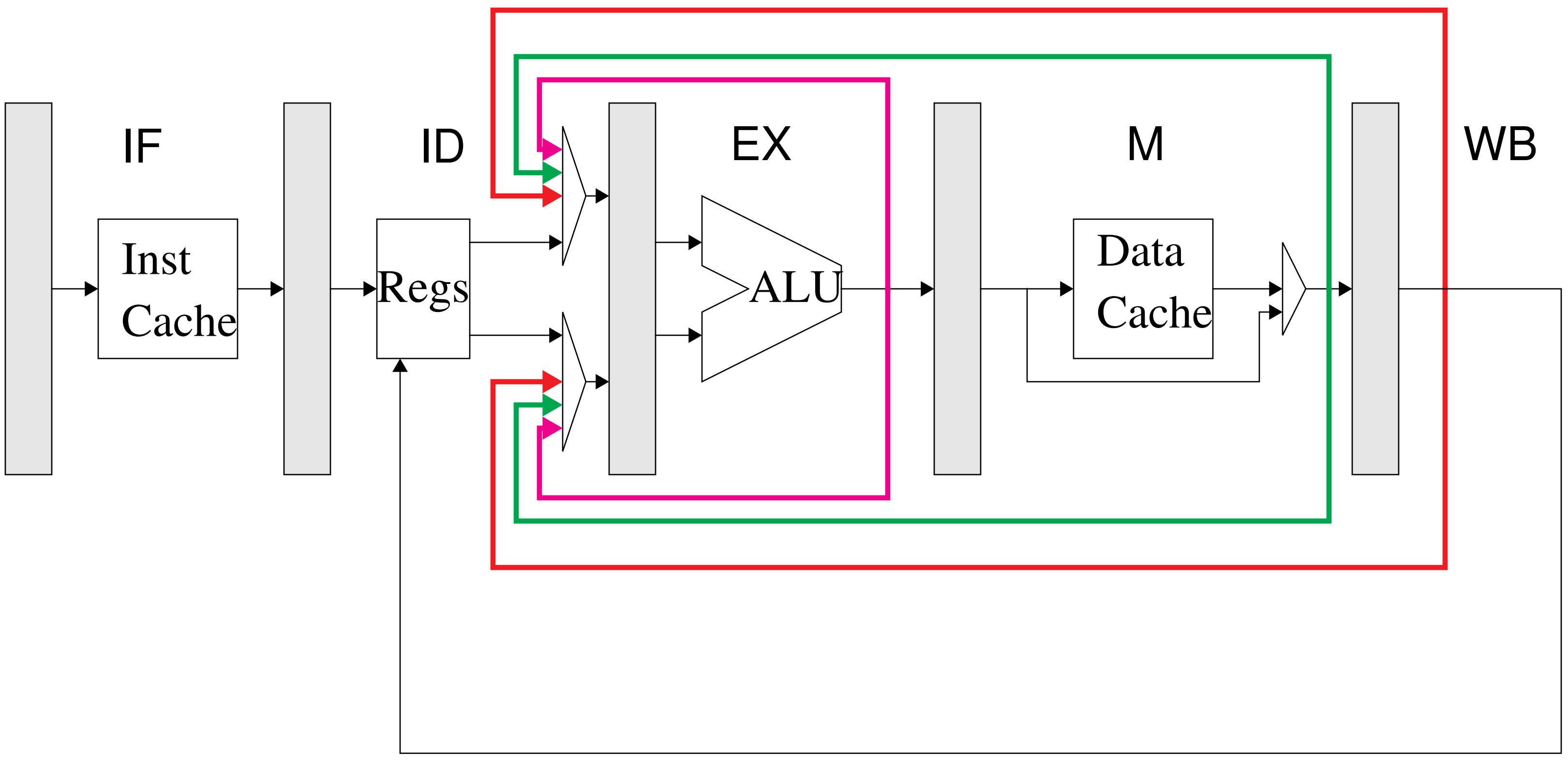
The Performance Impact of Incomplete Bypassing in Processor Pipelines

Pritpal S. Ahuja, Douglas W. Clark, Anne Rogers

Princeton University

A Pipeline with Bypassing



‘What do you mean “why’s it got to be built?” It’s a bypass. You’ve got to build bypasses.’

- Douglas Adams, *The Hitch Hiker’s Guide to the Galaxy*

Incomplete bypassing -- Why?

- Wide buses and muxes can consume much wiring area and power.
- Bypassing logic and control logic can pressure cycle time.
- Multi-issue and/or deeply pipelines machines amplify these effects:
  - Alpha 21064 has 45 separate bypass paths
  - a machine with “p” identical pipelines each with “s” stages that bypass requires:

$$\boxed{2p^2s} \quad \text{bypasses}$$

- It might be impractical to fully bypass a very wide (i.e. 8-issue) processor

Goals of this Work

- To measure the performance loss when various bypasses are removed.
- To measure how much of the lost performance can be recaptured by using software techniques.
- To gain some intuition as to the performance of more complex processors (VLIW, superscalar) where full bypassing may not be possible or desirable.

Previous Work

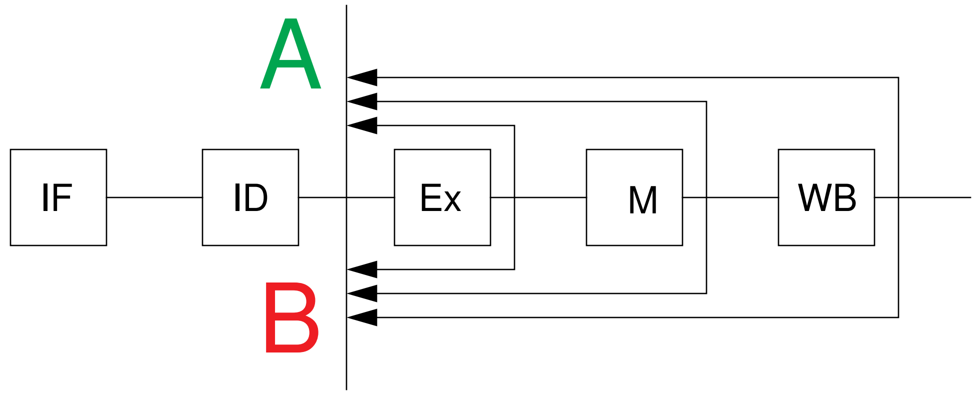
- Cohn, Gross, Lam, Tseng [1989]
  - iWarp processor running fp kernels
  - code scheduling reduced impact of missing bypasses
  - only studied a single configuration of missing bypasses
- Capitanio, Dutt, Nicolau [1992]
  - partitioned register files for VLIW machines
  - fewer register ports --> faster cycle time
- Abnous, Bagherzadeh [1994]
  - VIPER processor (VLIW)
  - incomplete bypassing between functional units

Outline

- ■ Background
- ❑ Experimental framework
- ❑ Experiments with incomplete bypassing
  - no optimization
  - commutative swapping
  - code scheduling
- ❑ Conclusions

Experimental Framework

- Machine model based on the MIPS R3000/R3100



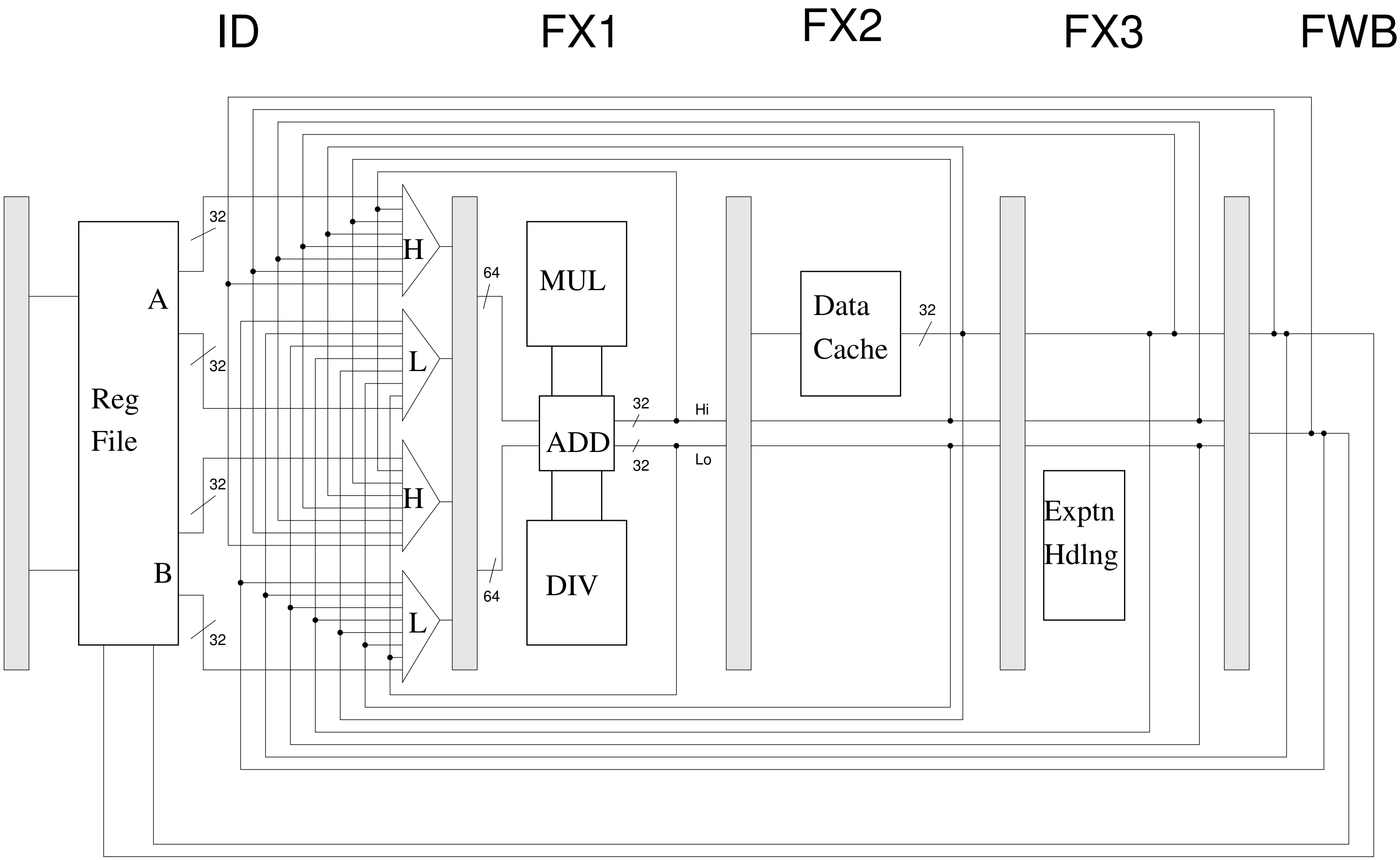
opcode $dest, $src1, $src2

add $3, $4, $5

load $1, 4($2)

slli $1, $2, 30

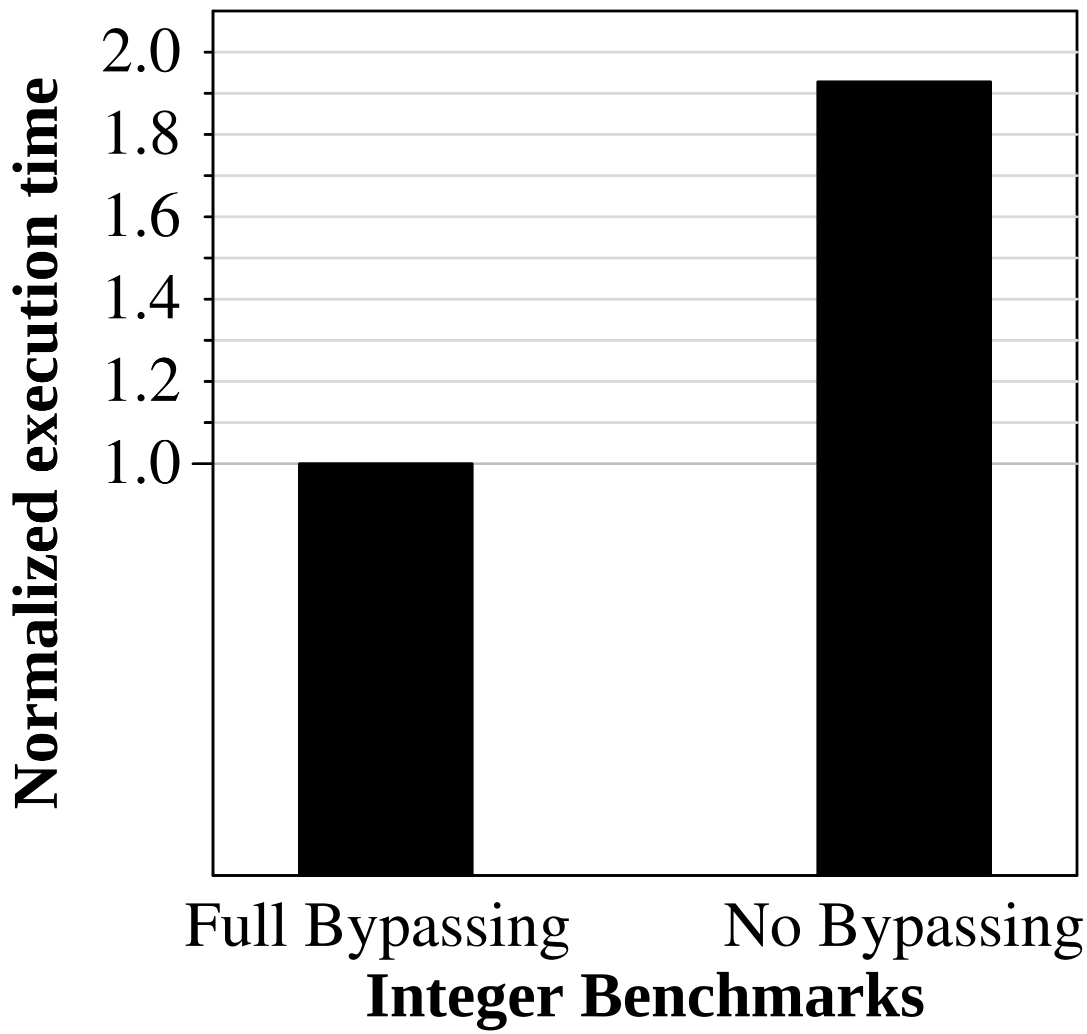
MIPS R3100 Floating-Point Pipeline



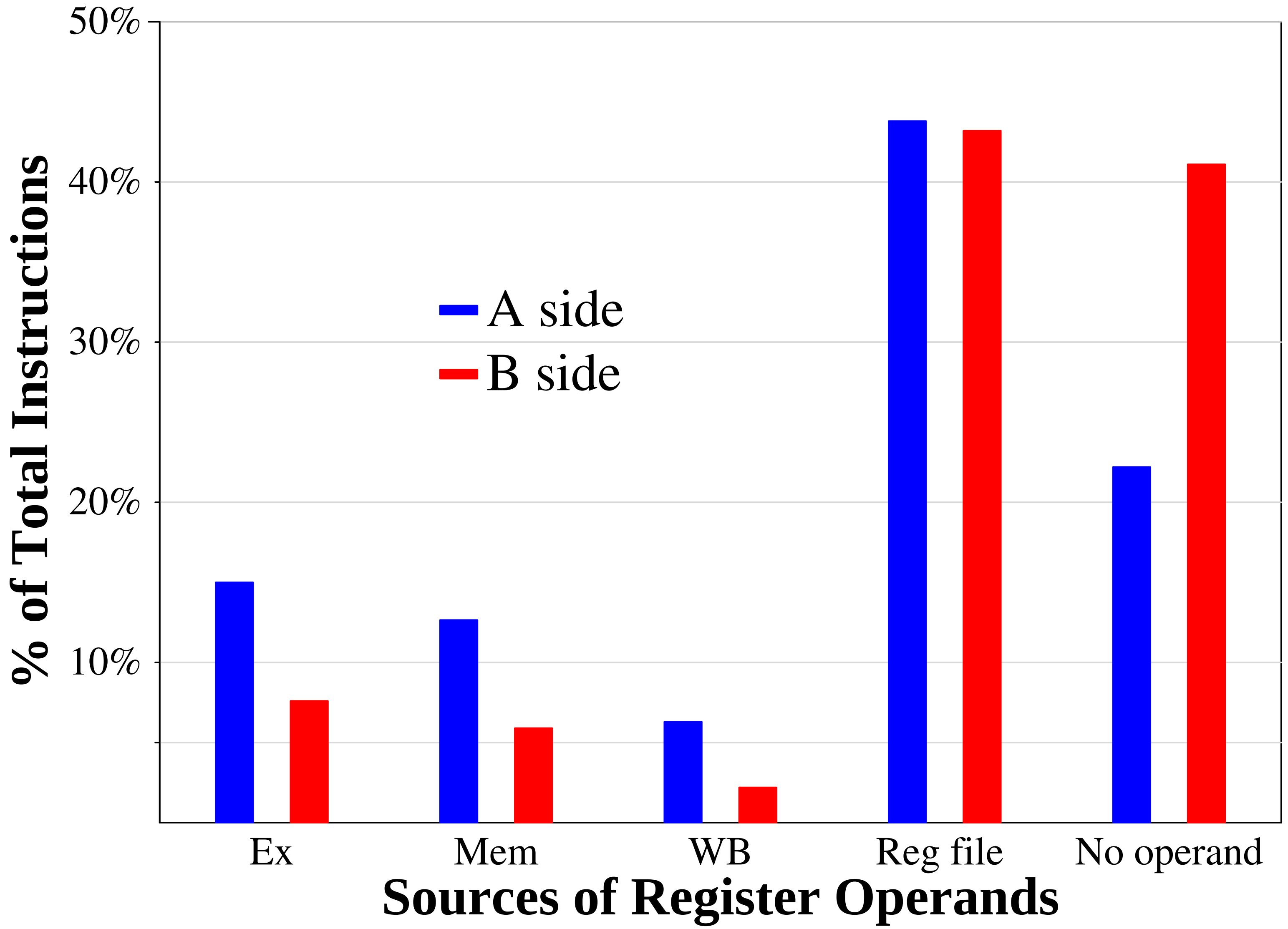
Experimental Framework (cont.)

- Cycle-level simulations with modified version of CL-SPIM
- 4 integer and 4 fp benchmarks from SPEC92
  - compress, eqntott, espresso, sc
  - doduc, hydro90, swm256, tomcatv
- Perfect caches (infinite size, 100% hit rate)
- Did not model clock cycle times of different configurations

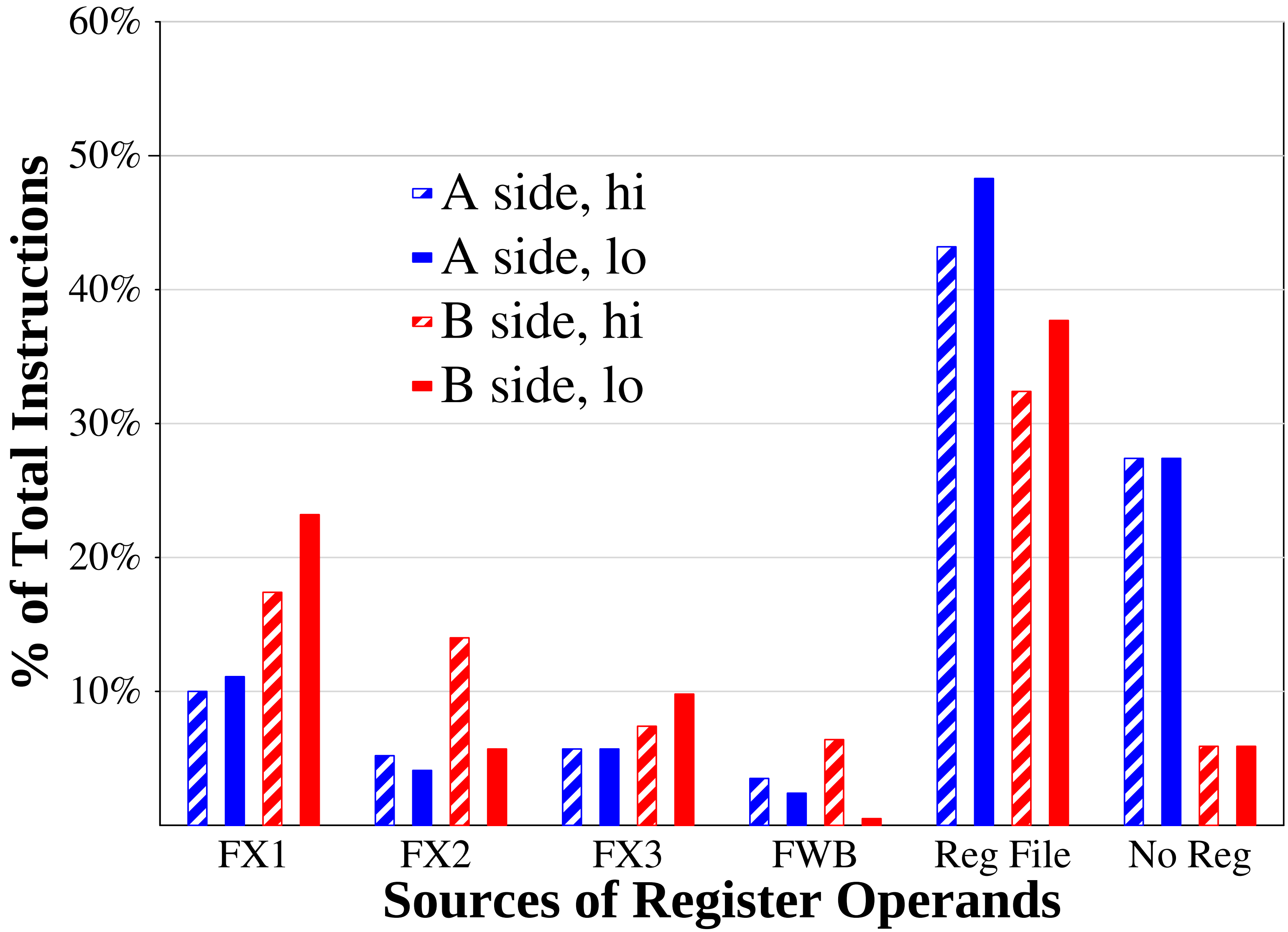
Bypassing's Effect on Performance



Integer Bypass Usage



Floating-Point Bypass Usage

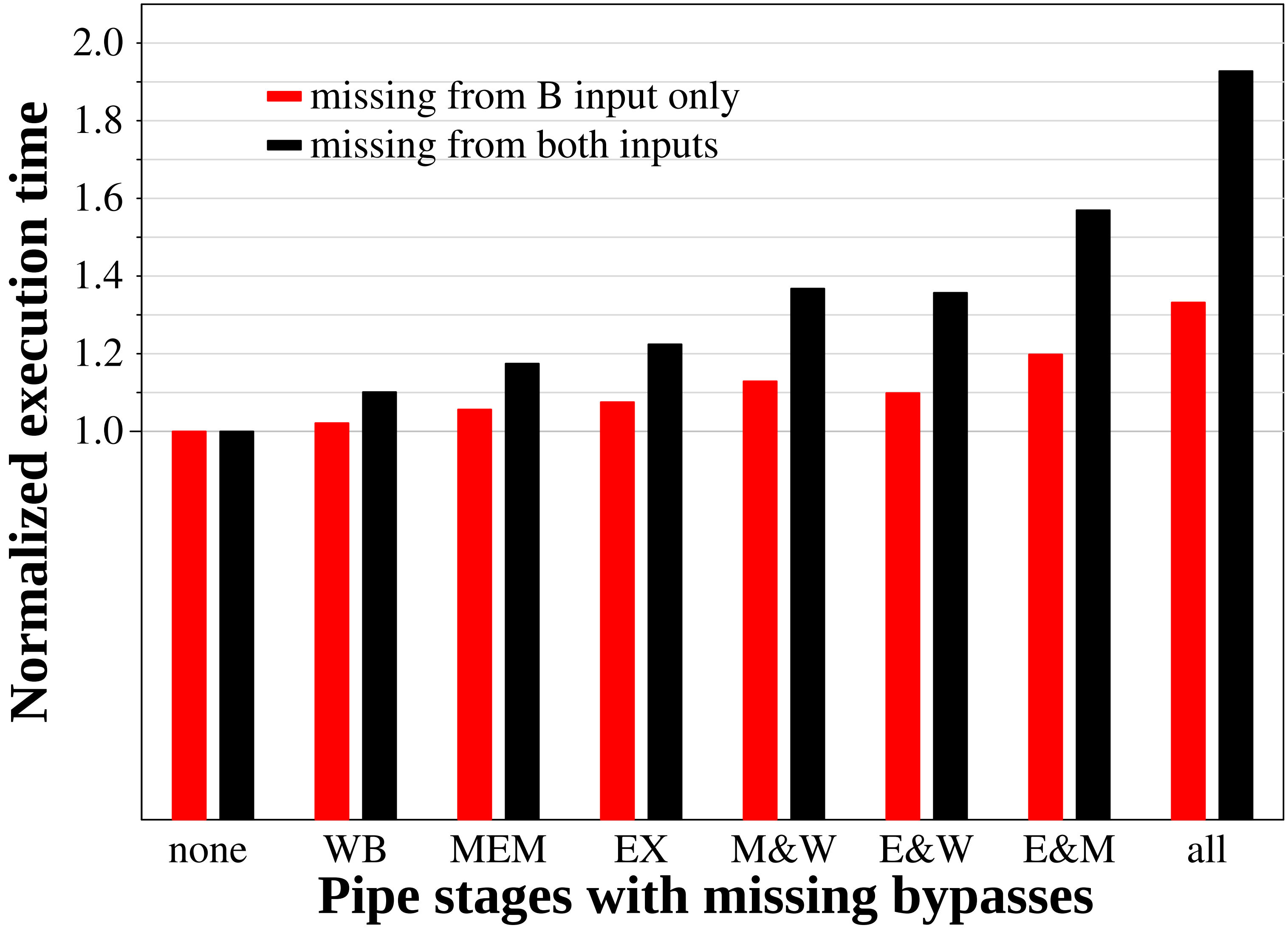


Experiments with Incomplete Bypassing

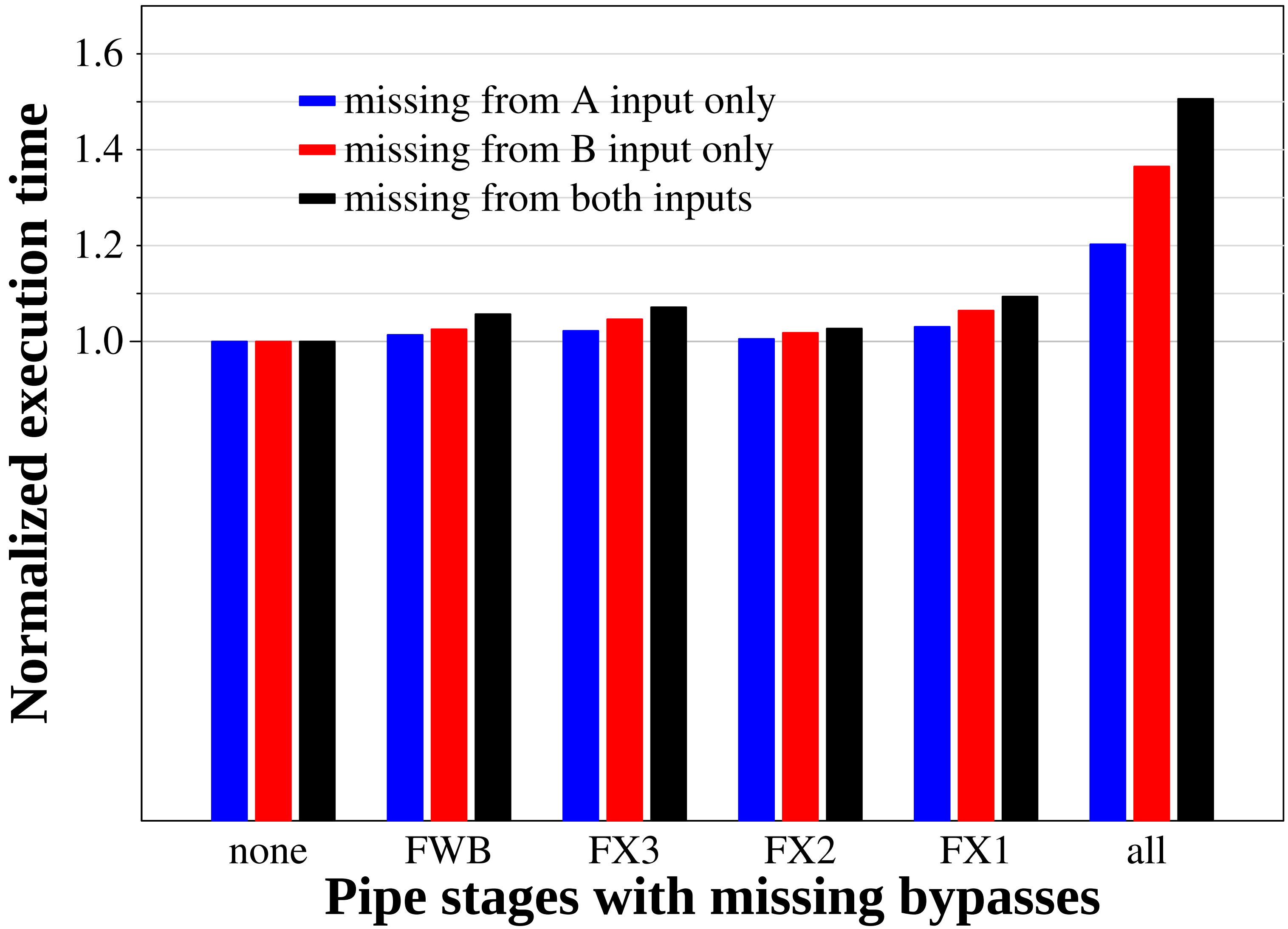
- Integer pipeline:
  - removed all combinations of 1, 2, and 3 bypasses from:
    - B-side,
    - both A-side and B-side
  - used the 4 SPECint benchmarks

- Floating-Point pipeline:
  - removed bypasses from 1 stage at a time:
    - A-side
    - B-side
    - both A-side and B-side
  - used the 4 SPECfp benchmarks

Incomplete Bypassing - Integer



Incomplete Bypassing - Floating-Point

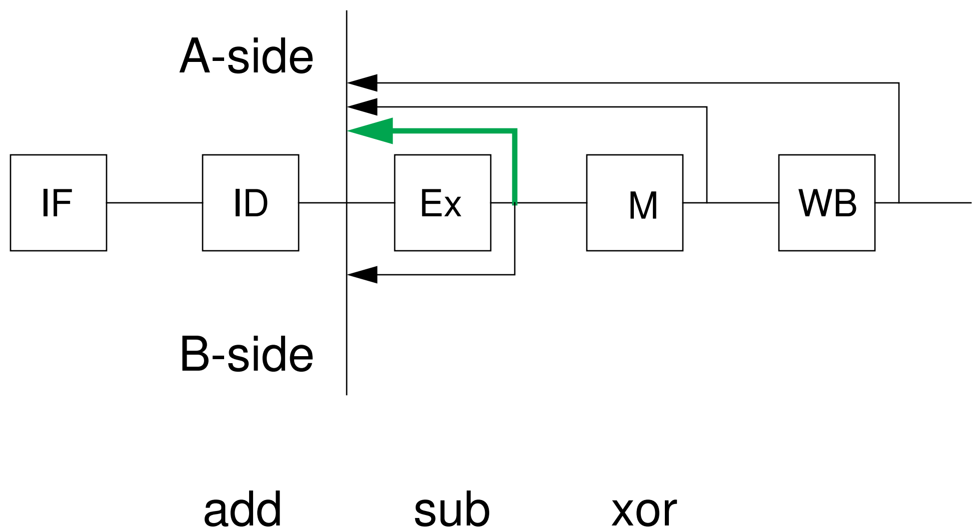


Commutative Swapping

xor $1, $2, $3

sub $4, $5, $6

add $7, $4, $1

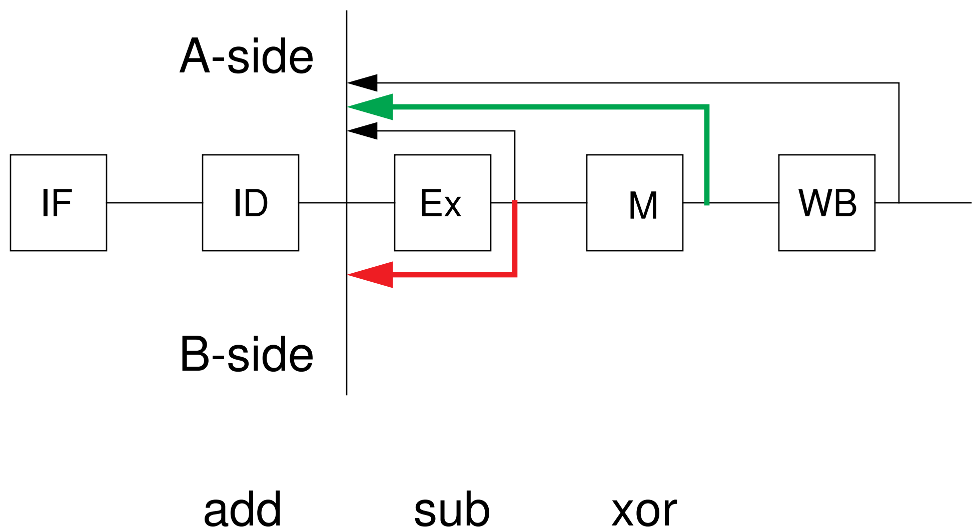


Commutative Swapping

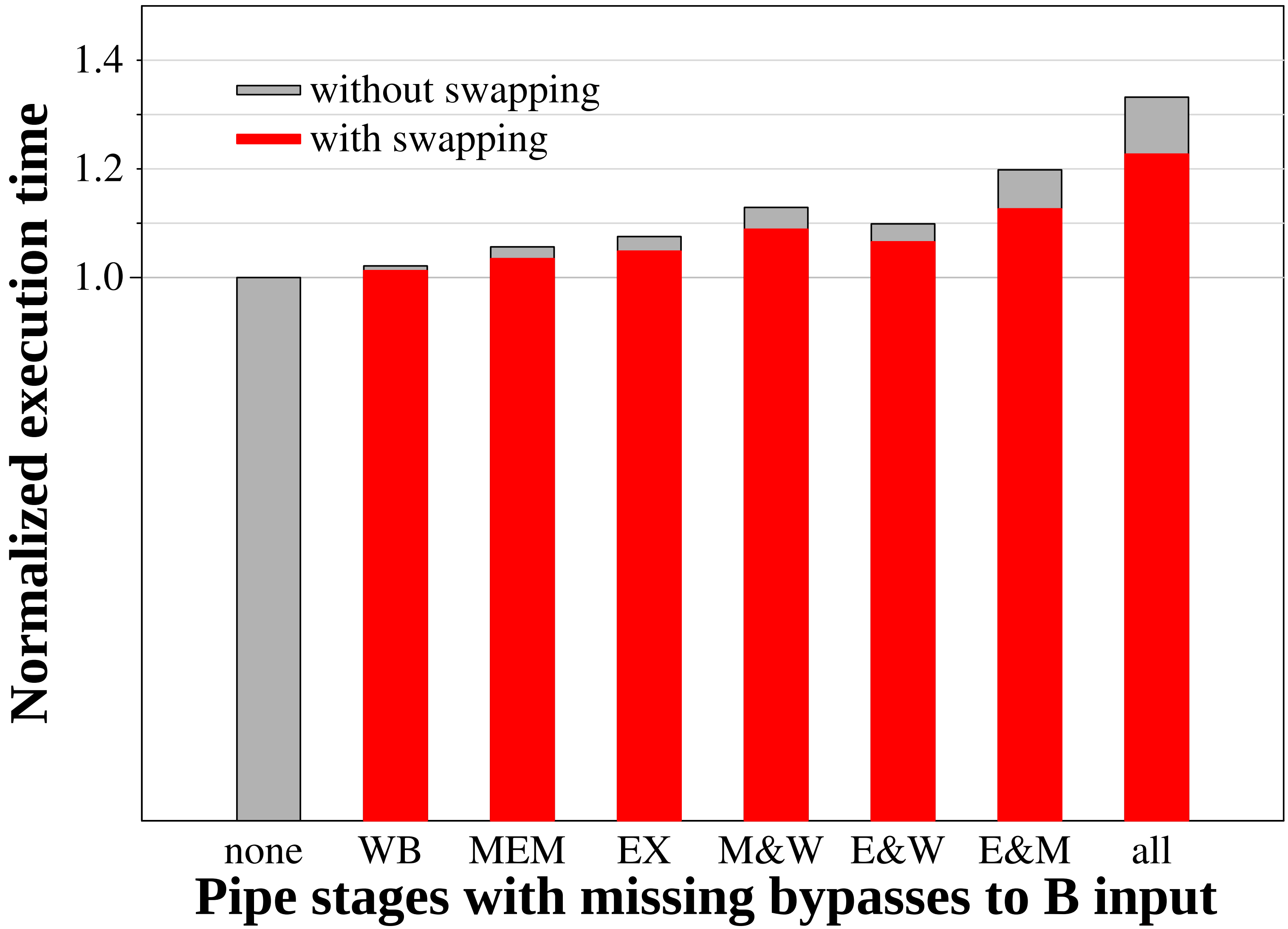
xor $1, $2, $3

sub $4, $5, $6

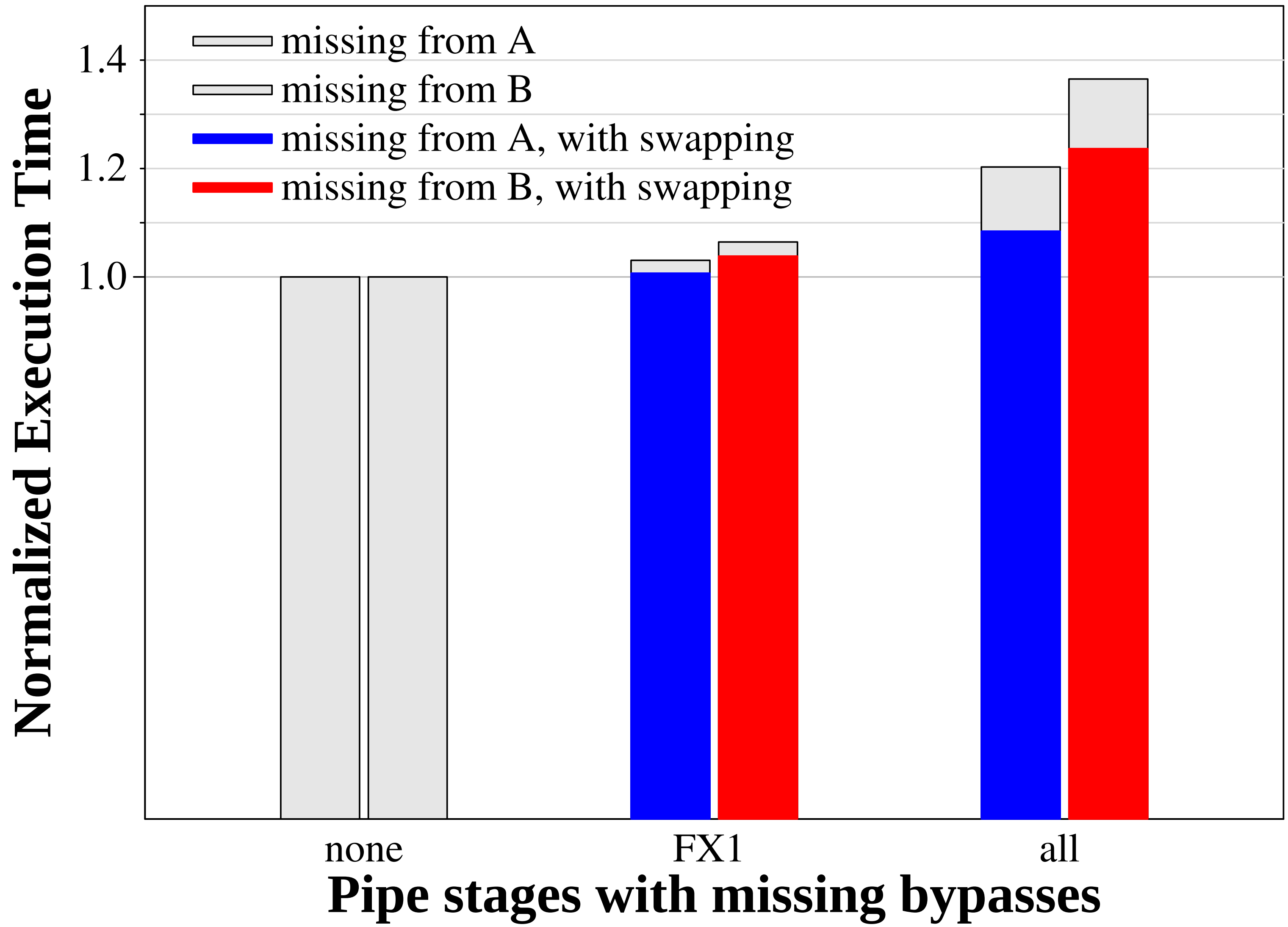
add $7, $1, $4



Asymmetric Bypassing - Integer

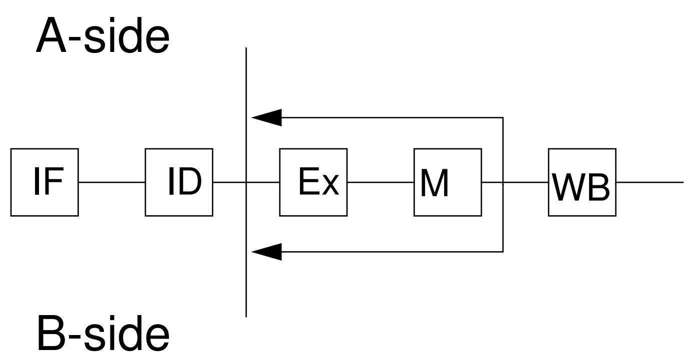


Asymmetric Bypassing - Floating-Point



Scheduling for Missing Bypasses

- Modified Gibbons and Muchnick's DAG-based scheduling algorithm to take irregular bypass structures into account



xor $1, $1, $1

sub$3, $1, $2

add $5, $6, $7

and$8, $9, $10

reschedule →

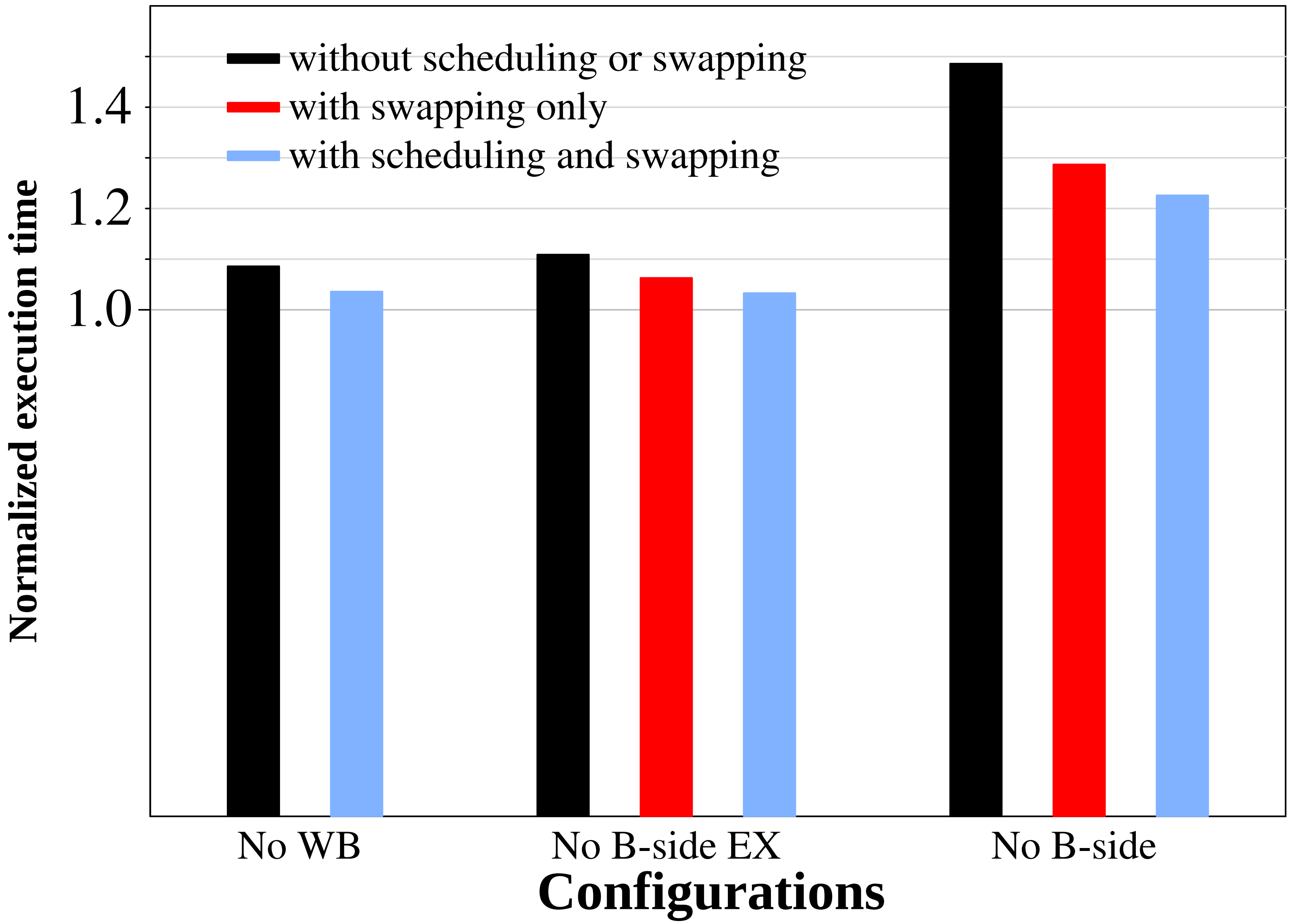
xor $1, $1, $1

add $5, $6, $6

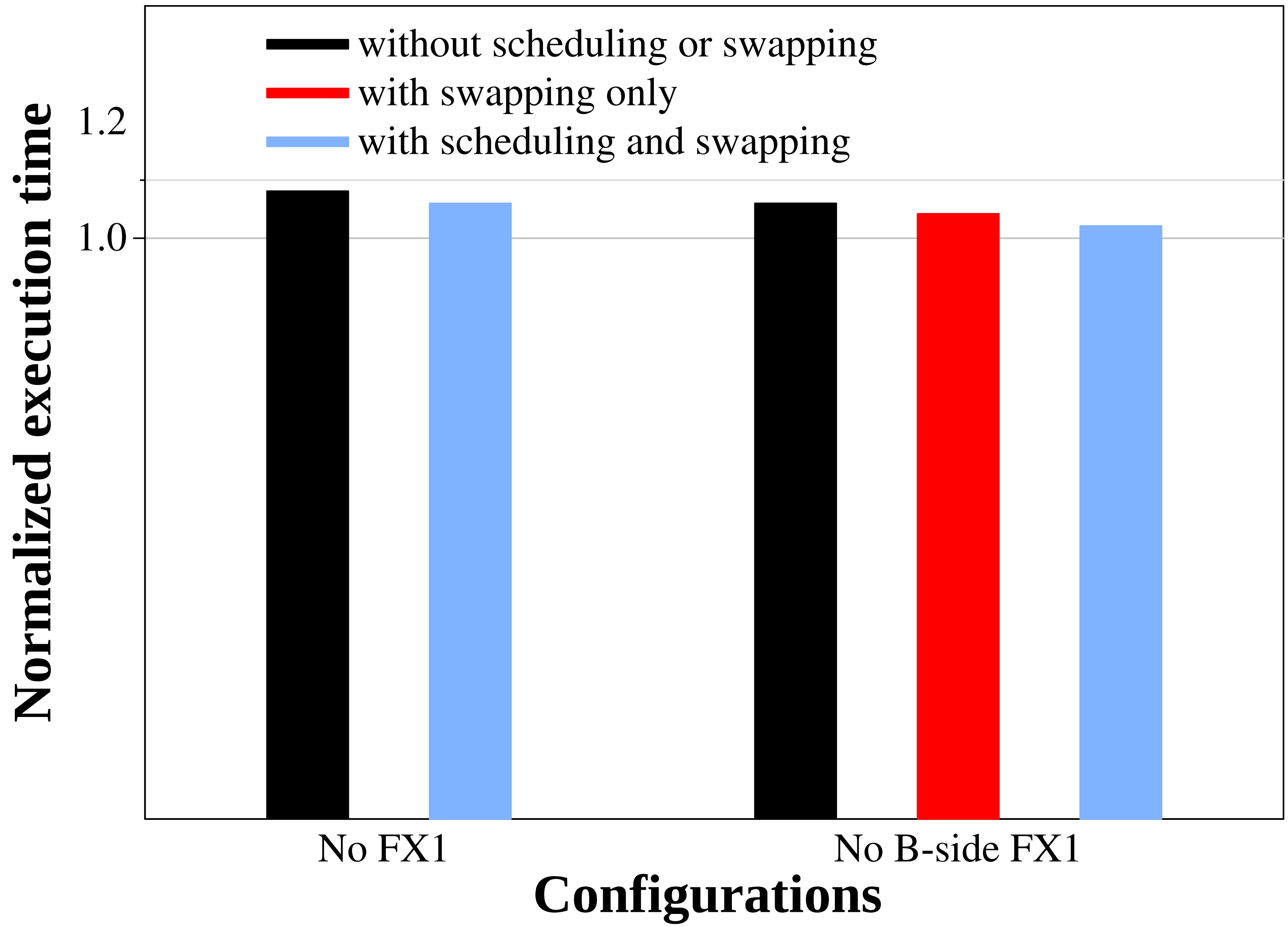
sub $3, $1, $2

and $8, $9, $10

Code Scheduling Experiments -- Compress



Code Scheduling Experiments -- Tomcatv



Conclusions

- Incomplete bypassing can degrade performance from a few percent to over 90 percent
- Commutative swapping can recapture 20%-50% of the lost performance
- Code scheduling can eliminate many of the remaining interlock stalls

"Shut up and go away, and take your bloody bypass with you."

- Douglas Adams, *The Hitch Hiker's Guide to the Galaxy*