Lecture 7
Instruction Level Parallelism (5)

EEC 171 Parallel Architectures

John Owens
UC Davis

Credits

- © John Owens / UC Davis 2007–2009.
- Thanks to many sources for slide material: Computer Organization and Design (Patterson & Hennessy) © 2005, Computer Architecture (Hennessy & Patterson) © 2007, Inside the Machine (Jon Stokes) © 2007, © Dan Connors / University of Colorado 2007, © Wen-Mei Hwu/David Kirk, University of Illinois 2007, © David Patterson / UCB 2003–7, © John Lazzaro / UCB 2006, © Mary Jane Irwin / Penn State 2005, © John Kubiatowicz / UCB 2002, © Krste Asinovic/Arvind / MIT 2002, © Morgan Kaufmann Publishers 1998.

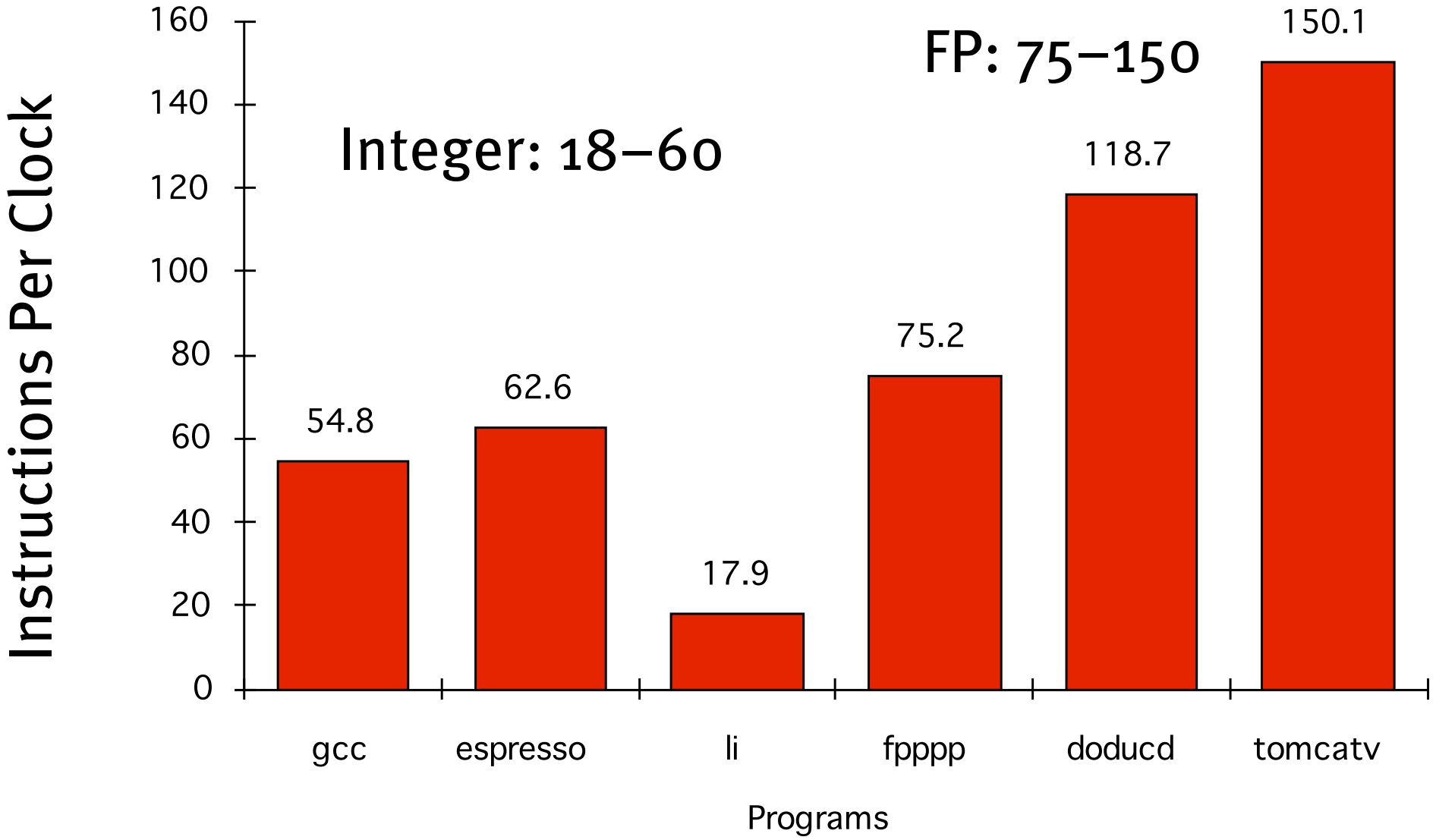
Limits to ILP

- Conflicting studies of amount
  - Benchmarks (vectorized Fortran FP vs. integer C programs)
  - Hardware sophistication
  - Compiler sophistication
- How much ILP is available using existing mechanisms with increasing HW budgets?
- Do we need to invent new HW/SW mechanisms to keep on processor performance curve?
  - Intel MMX, SSE (Streaming SIMD Extensions): 64 bit ints
  - Intel SSE2: 128 bit, including 2 64-bit Fl. Pt. per clock
  - Motorola AltiVec: 128 bit ints and FPs
  - Supersparc Multimedia ops, etc.

Overcoming Limits

- Advances in compiler technology + significantly new and different hardware techniques may be able to overcome limitations assumed in studies
- However, unlikely such advances when coupled with realistic hardware will overcome these limits in near future

Upper Limit to ILP: Ideal Machine



Limits to ILP

- Most techniques for increasing performance increase power consumption
- The key question is whether a technique is energy efficient: does it increase power consumption faster than it increases performance?
- Multiple issue processor techniques all are energy inefficient:
  - Issuing multiple instructions incurs some overhead in logic that grows faster than the issue rate grows
  - Growing gap between peak issue rates and sustained performance
- Number of transistors switching = f(peak issue rate), and performance = f(sustained rate):
  Growing gap between peak and sustained performance
  ⇒ increasing energy per unit of performance

Limits to ILP

- Doubling issue rates above today's 3–6 instructions per clock, say to 6 to 12 instructions, probably requires a processor to
  - Issue 3 or 4 data memory accesses per cycle,
  - Resolve 2 or 3 branches per cycle,
  - Rename and access more than 20 registers per cycle, and
  - Fetch 12 to 24 instructions per cycle.
- Complexities of implementing these capabilities likely means sacrifices in maximum clock rate
  - E.g, widest issue processor is the Itanium 2, but it also has the slowest clock rate, despite the fact that it consumes the most power!

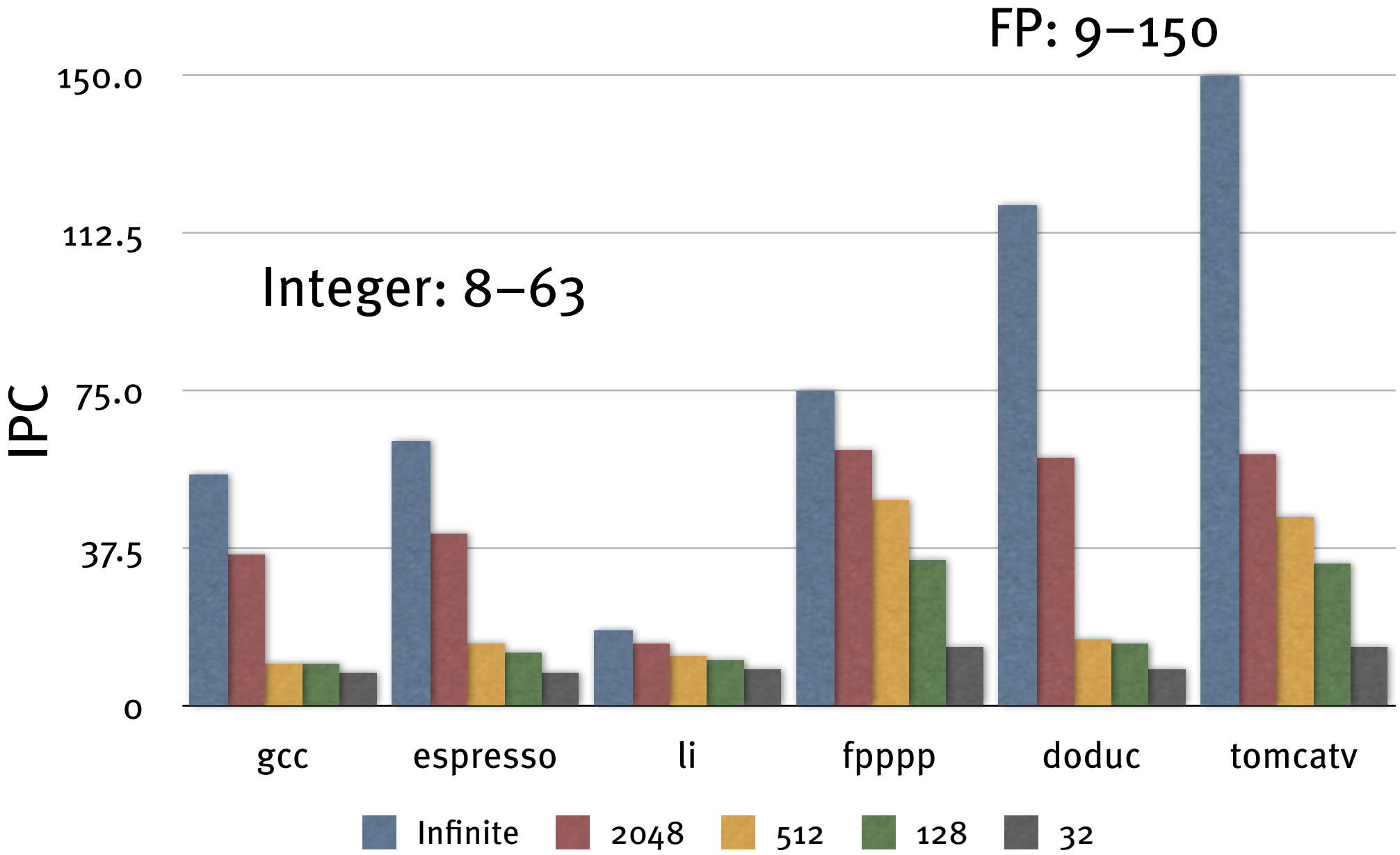
Limits to ILP

- Initial HW Model here; MIPS compilers.
- Assumptions for ideal/perfect machine to start:
  - 1. Register renaming – infinite virtual registers
    => all register WAW & WAR hazards are avoided
  - 2. Branch prediction – perfect; no mispredictions
  - 3. Jump prediction – all jumps perfectly predicted (returns, case statements)
    2 & 3 ⇒ no control dependencies; perfect speculation & an unbounded buffer of instructions available
  - 4. Memory-address alias analysis – addresses known & a load can be moved before a store provided addresses not equal; 1&4 eliminates all but RAW
- Also: perfect caches; 1 cycle latency for all instructions (FP *,/); unlimited instructions issued/clock cycle;

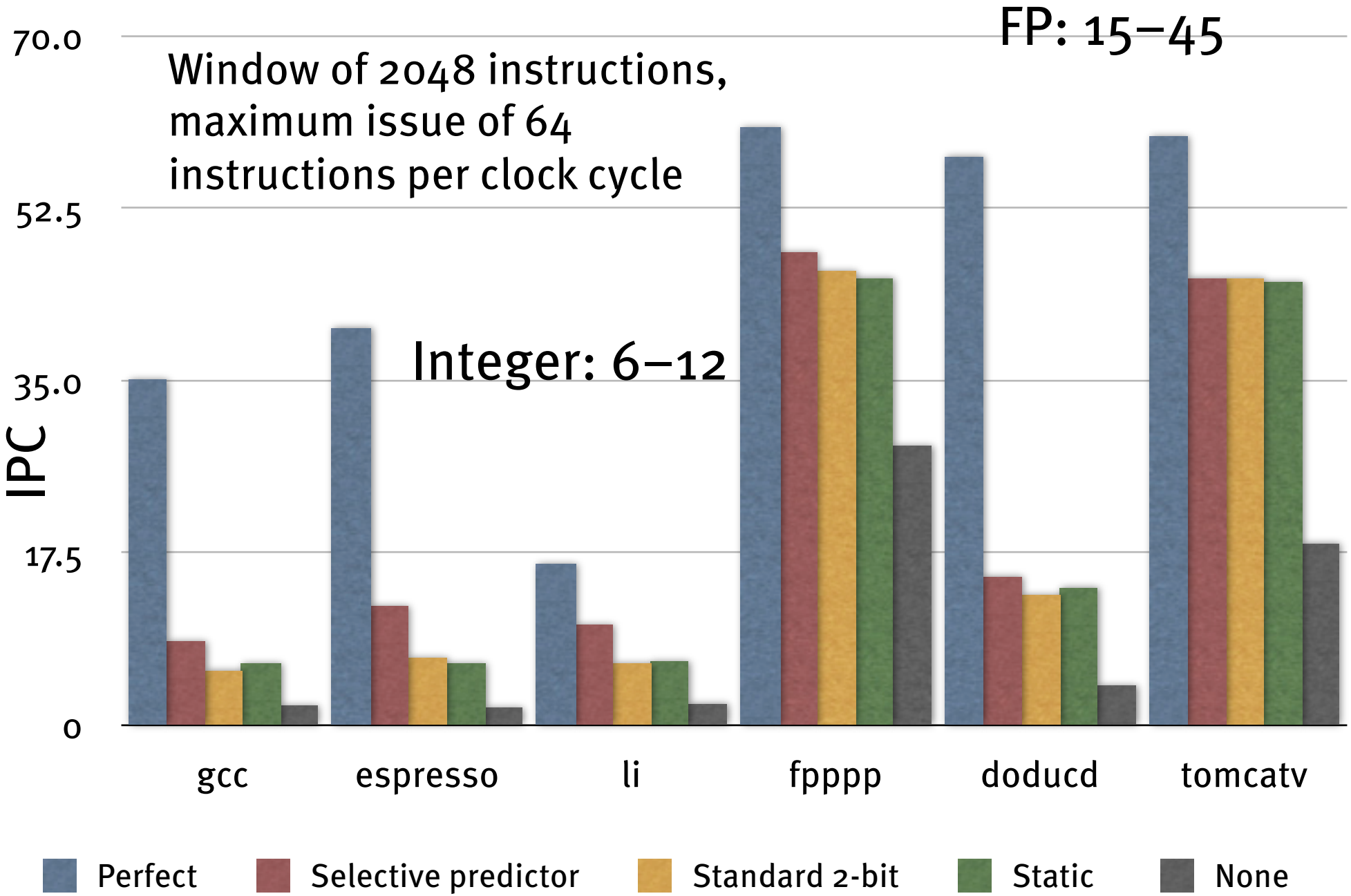
Limits to ILP HW Model comparison

	New Model	Ideal	Power 5
Instructions Issued per clock	64	Infinite	4
Instruction Window Size	2048	Infinite	200
Renaming Registers	256 Int + 256 FP	Infinite	48 integer + 40 Fl. Pt.
Branch Prediction	8K 2-bit	Perfect	Tournament
Cache	Perfect	Perfect	64KI, 32KD, 1.92MB L2, 36 MB L3
Memory Alias	Perfect v. Stack v. Inspect v. none	Perfect	Perfect

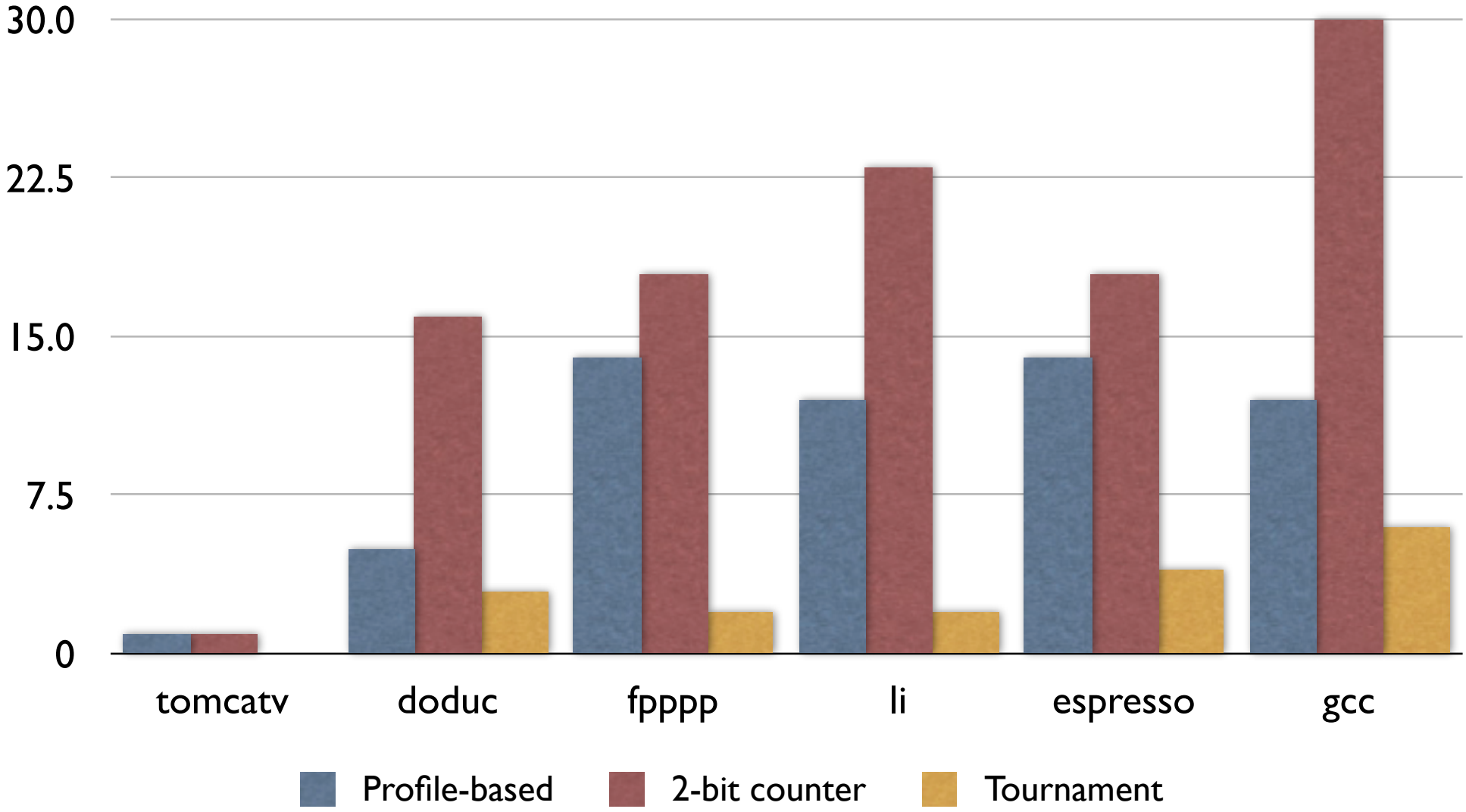
More Realistic HW: Window Impact



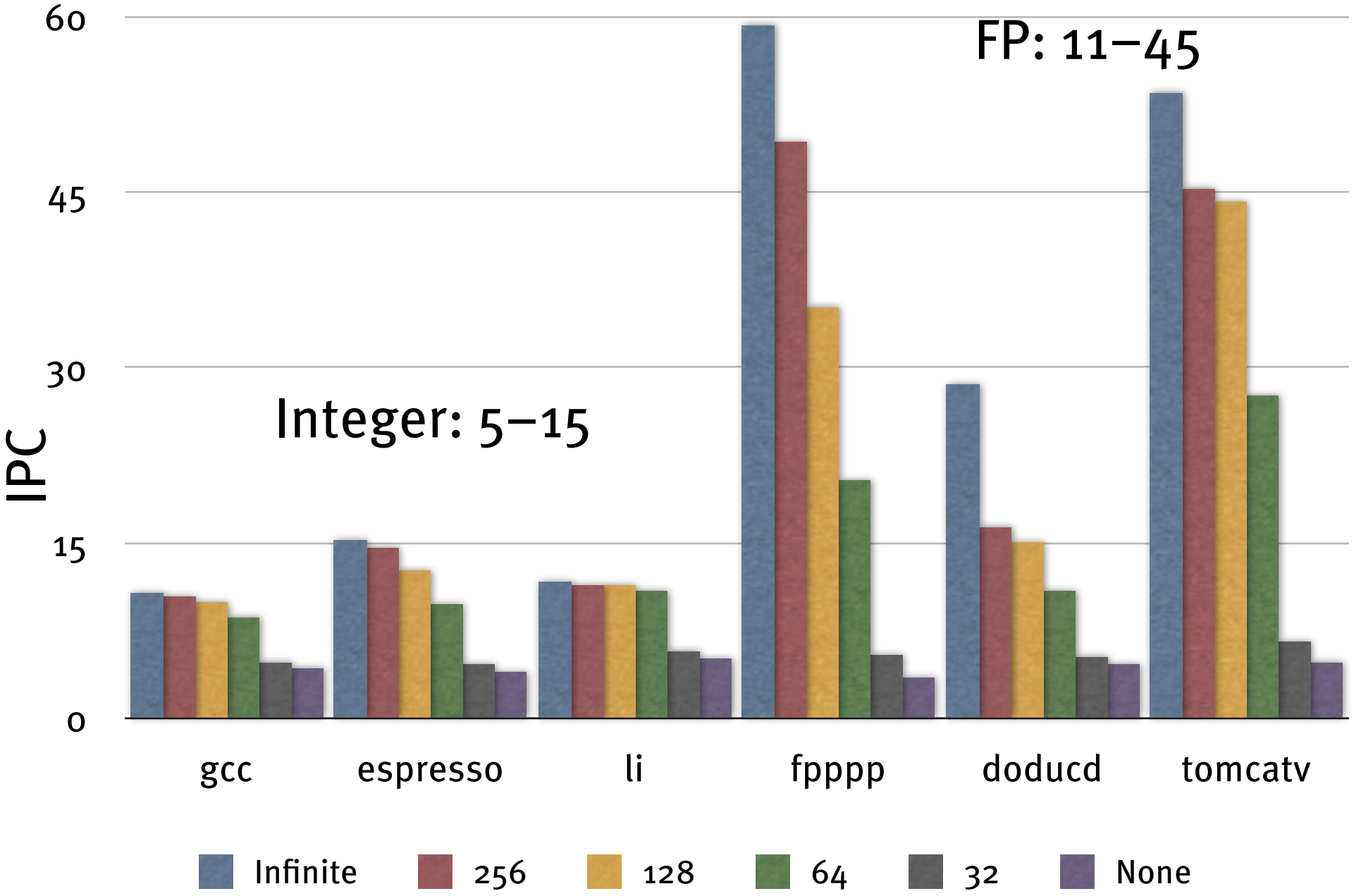
More Realistic HW: Branch Impact



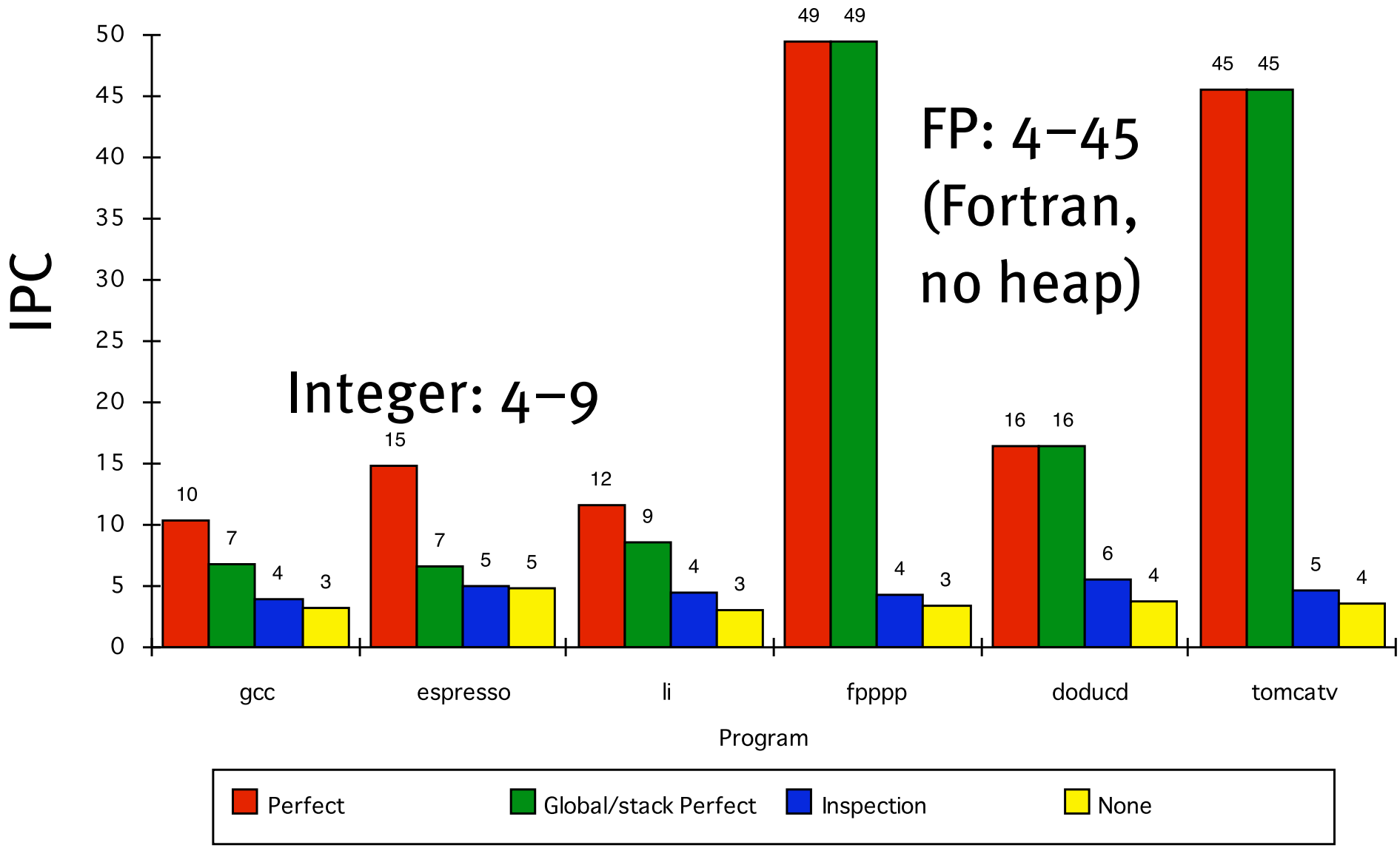
Misprediction Rates (%)



Renaming Register Impact



Memory Address Alias Impact



HW vs. SW to increase ILP

- Memory disambiguation: HW best
- Speculation:
  - HW best when dynamic branch prediction better than compile time prediction
  - Exceptions easier for HW
  - HW doesn't need bookkeeping code or compensation code
  - Very complicated to get right
- Scheduling: SW can look ahead to schedule better
- Compiler independence: does not require new compiler, recompilation to run well

Performance beyond single thread ILP

- There can be much higher natural parallelism in some applications (e.g., Database or Scientific codes)
- Explicit Thread Level Parallelism or Data Level Parallelism
- Thread: process with own instructions and data
  - thread may be a process part of a parallel program of multiple processes, or it may be an independent program
  - Each thread has all the state (instructions, data, PC, register state, and so on) necessary to allow it to execute
- Data Level Parallelism: Perform identical operations on data, and lots of data

ILP Summary

- Leverage Implicit Parallelism for Performance: Instruction Level Parallelism
- Loop unrolling by compiler to increase ILP
- Branch prediction to increase ILP
- Dynamic HW exploiting ILP
  - Works when can't know dependence at compile time
  - Can hide L1 cache misses
  - Code for one machine runs well on another