Itanium™ Performance Insights

W. Hwu,

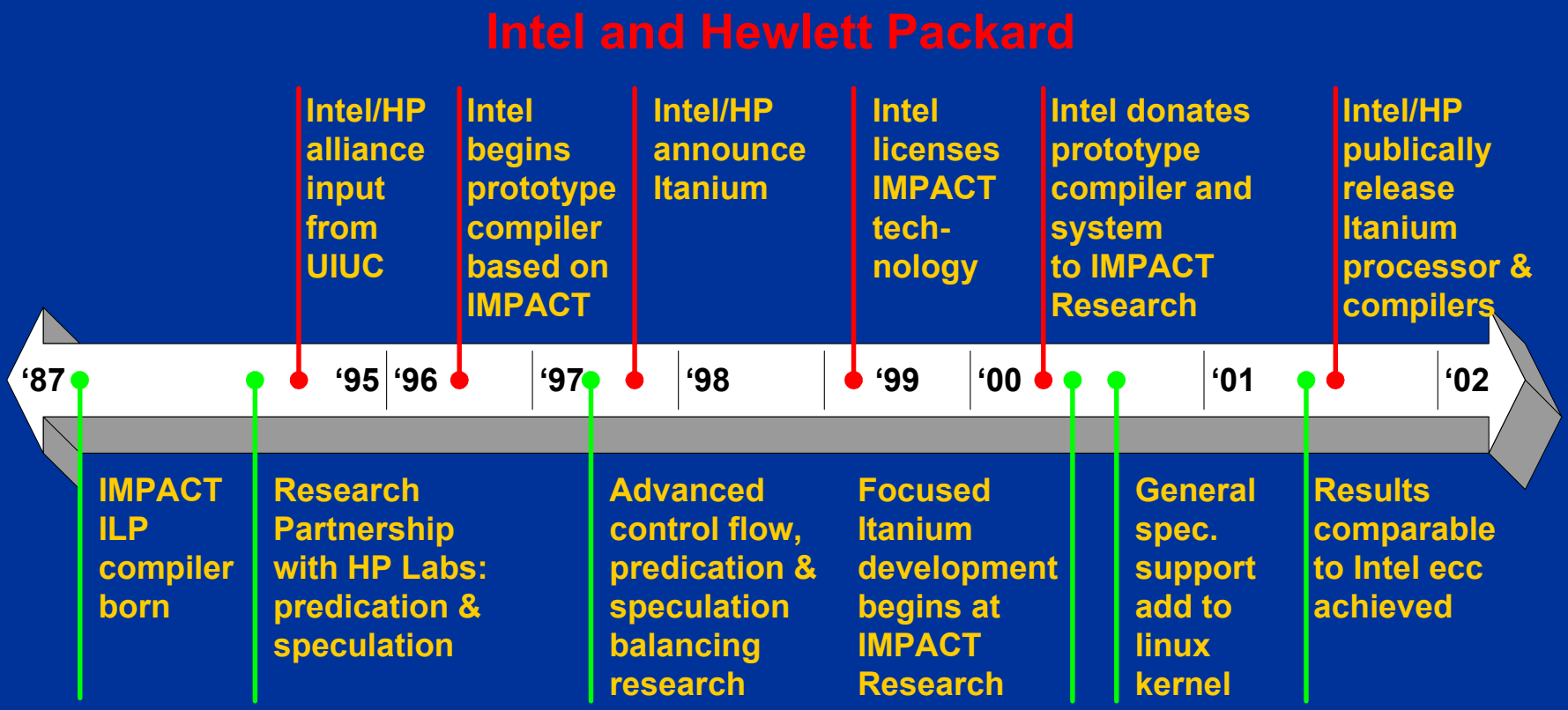
J. Sias, M. Merten, E. Nystrom, R. Barnes,
C. Shannon, S. Ryoo, J. Olivier

IMPACT Research Group
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
http://www.crhc.uiuc.edu/IMPACT/

Predictions from MPF 1998

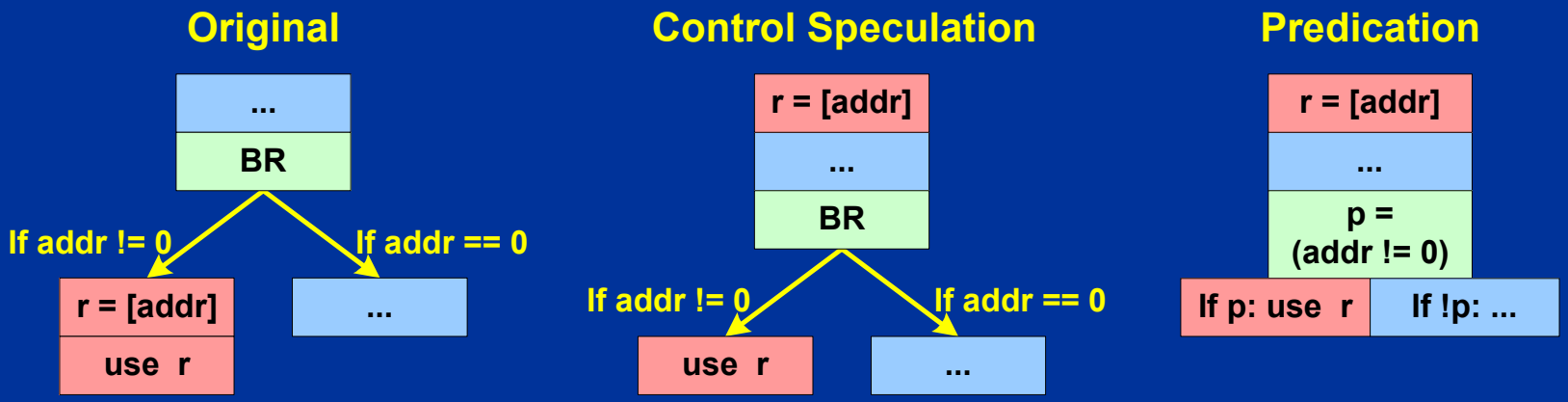
- Compilers critical to the performance of EPIC uP's
  - Use of predication and speculation is a serious challenge.
  - Any misuse will lead to performance loss.
  - Brand new algorithms will be deployed in the EPIC compilers.
  - Existing software development models must be supported.
- Expect performance robustness issues
  - Awesome performance leap seen for some applications.
  - Less for others due to limitations of analyses and optimizations.
  - It can take years for the performance gain to be universal.
- Evolution of EPIC architectures
  - Revisions of architectures are likely as compilers mature.
  - Code size and power consumption are critical for embedded EPICs.
- After three years…what is the reality?

Itanium Development History



Itanium Architecture Overview

- Itanium design goal: enhance scalability of parallelism by moving complex decisions to the compiler
  - Bundling: enables static scheduling by communicating instruction parallelism
  - Predication: allows compiler to optimize across multiple paths by providing an alternative to control flow
  - Speculation support: allows compiler to select specific instructions for early execution



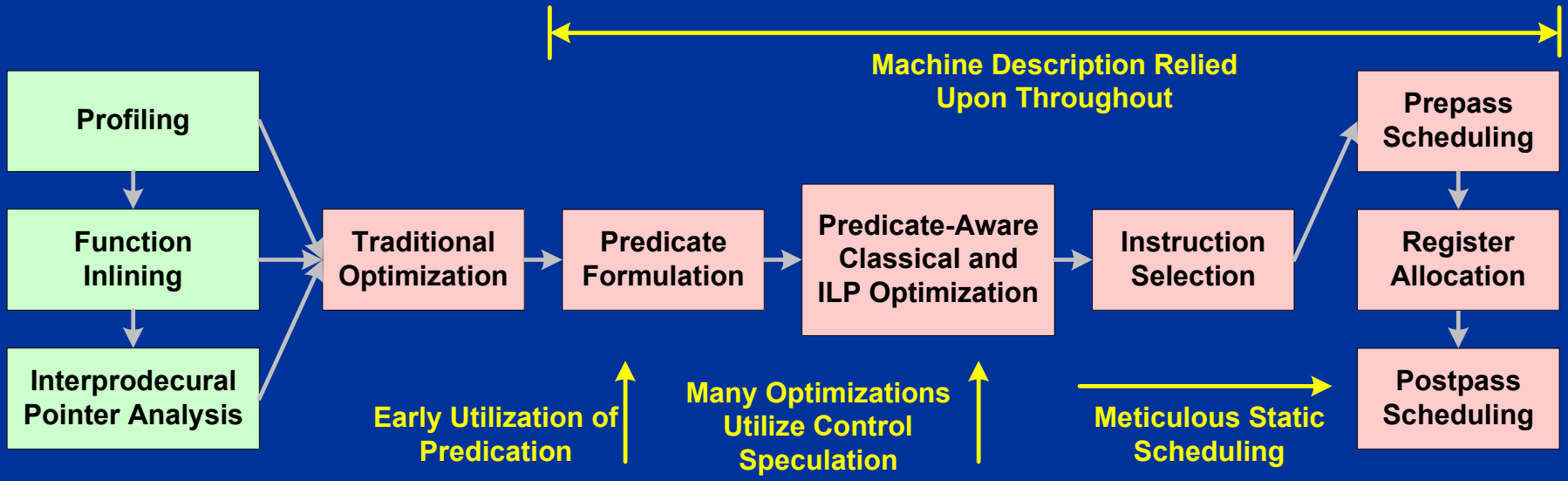
Itanium Compilation Landscape

- Increased reliance on the compiler for performance
  - Explicit control of the architecture: realities of modern microarchitecture have become visible at software level
  - Particular problems: effects of runtime uncertainty
    - Control resolution, variable memory latency, etc.
  - Solutions from EPIC/VLIW research
    - Memory disambiguation, profiling
    - Static scheduling, control speculation, predication

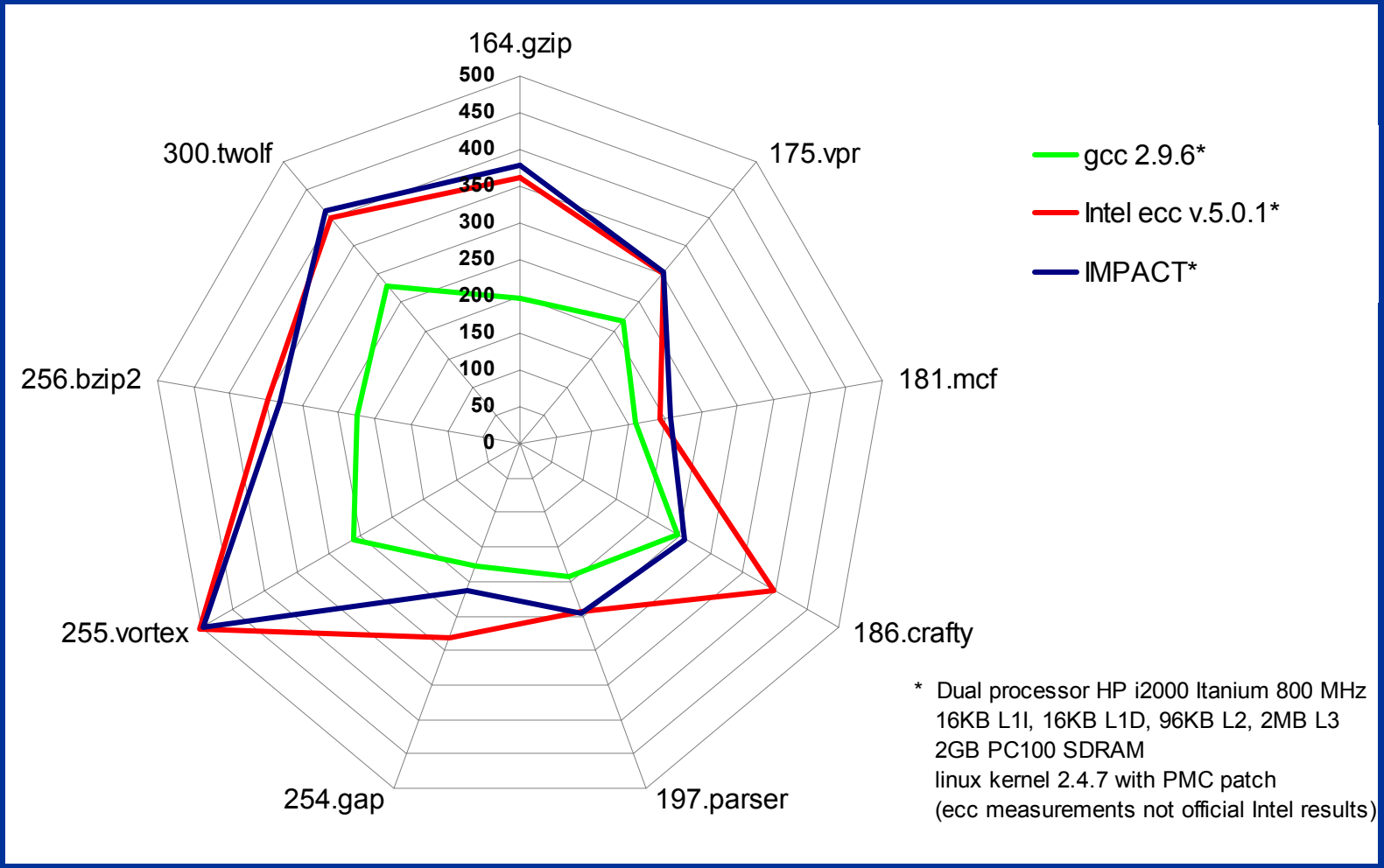
	Application coverage	Public/ Proprietary	Peephole opti level	ILP opti level	Extens-ibility	Mode
GNU gcc	Very High	Public	Low	Very Low	Low	LP64
Intel ecc	High	Proprietary	High	High	High	LP64
HP aCC	High	Proprietary	High	High	High	ILP32
IMPACT	Medium	Future Public	Medium	Very High	Very High	LP64

IMPACT Compiler

- IMPACT compiler supports ILP compilation and research
  - Extensible framework for easy implementation of new optimizations
  - Versatile and pervasive intermediate representation (IR)
  - Advanced interprocedural pointer analysis
  - Comprehensive predicate-aware dataflow and predicate analyses
  - Direct analysis from IR at most stages of compilation

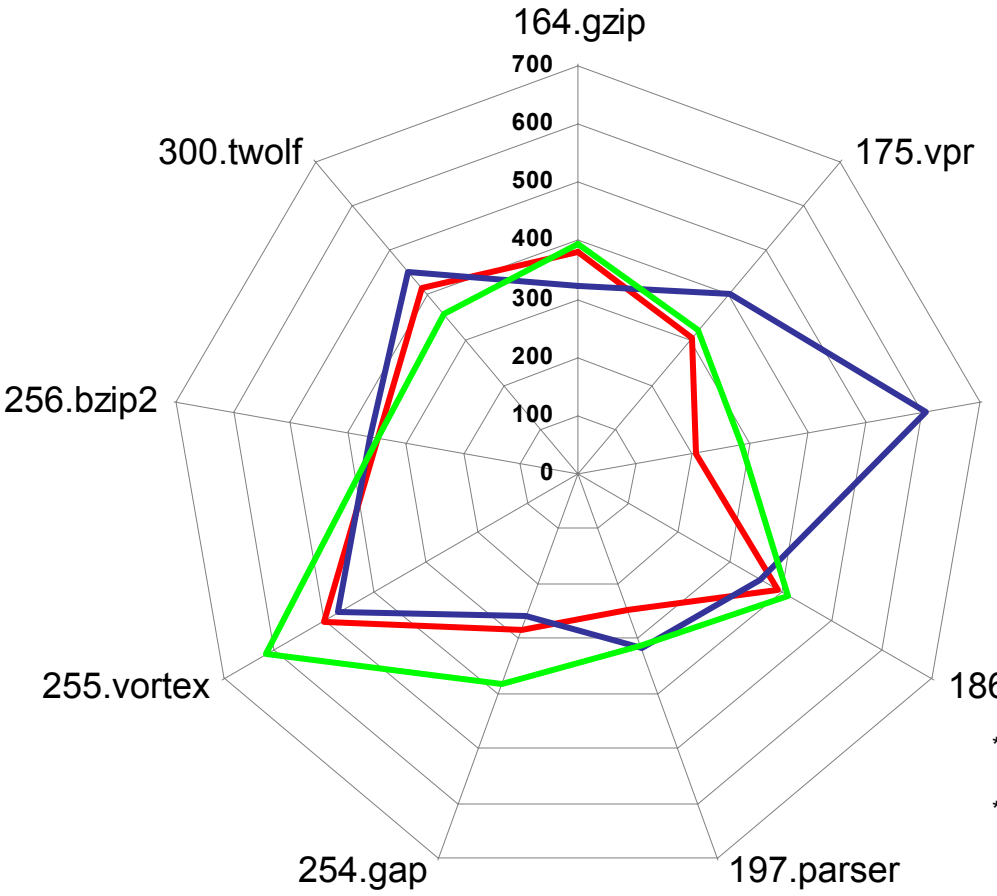


LP64 Compiler Comparison



SPECint2000

Cross-Platform Comparison



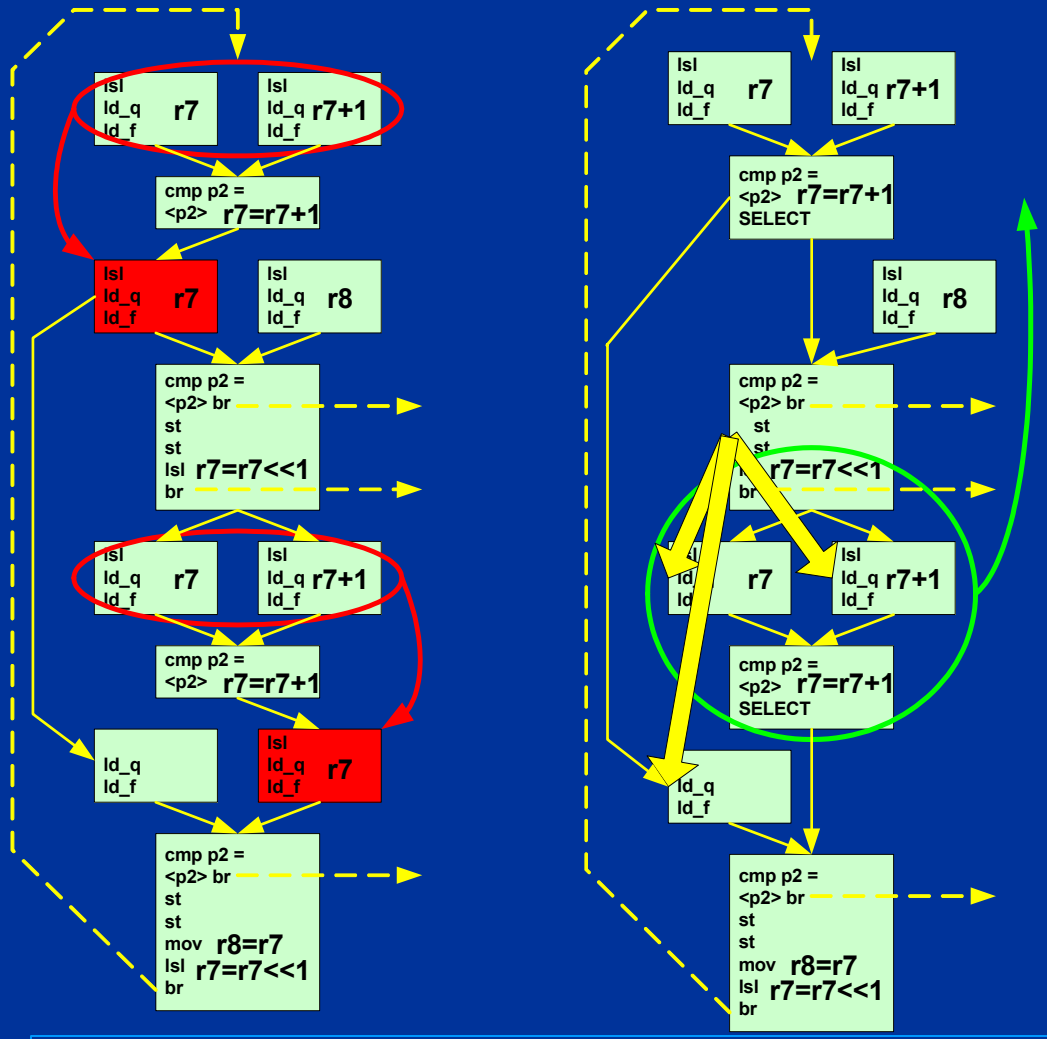
* Dual processor HP i2000 Itanium 800 MHz
16KB L1I, 16KB L1D, 96KB L2, 2MB L3
2GB PC100 SDRAM
linux kernel 2.4.7 with PMC patch
(ecc measurements not official Intel results)

** Source: spec.org. 4MB L3 cache

*** Source: spec.org. D815EEA2 MB
16KB L1I, 16KB L1D, 256KB L2
256MB PC133 CL2 SDRAM

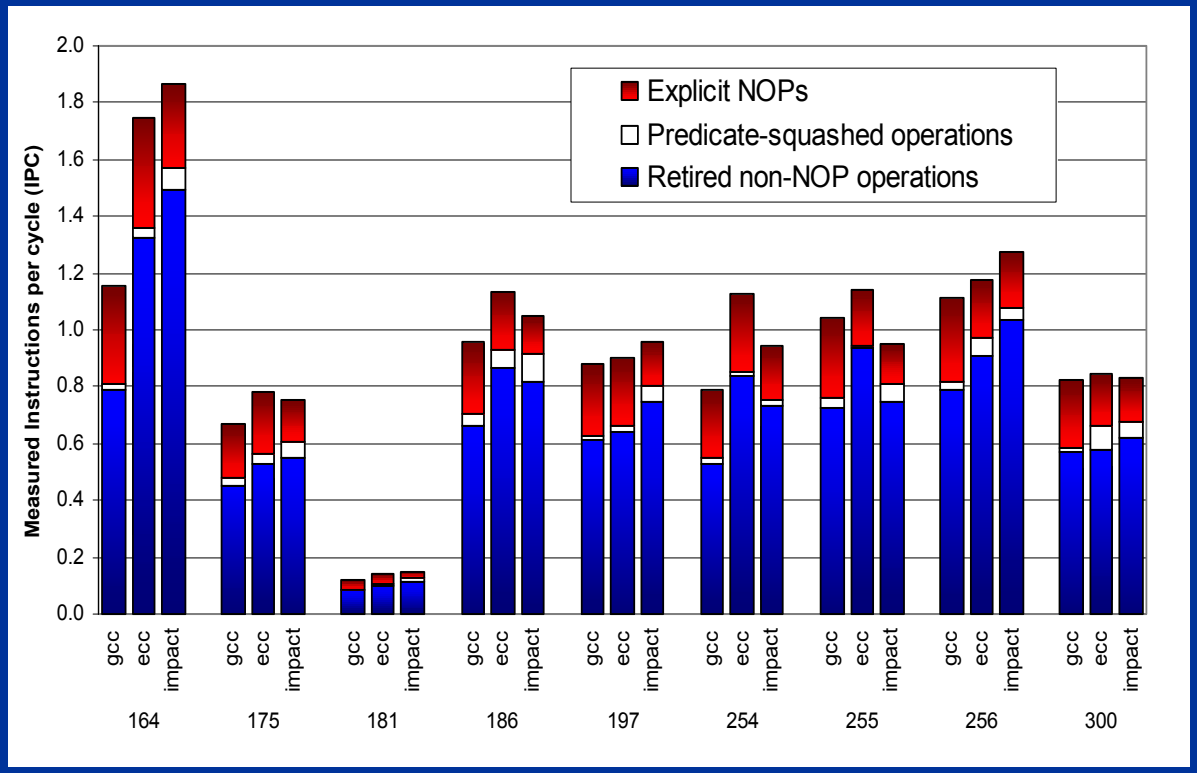
SPECint2000

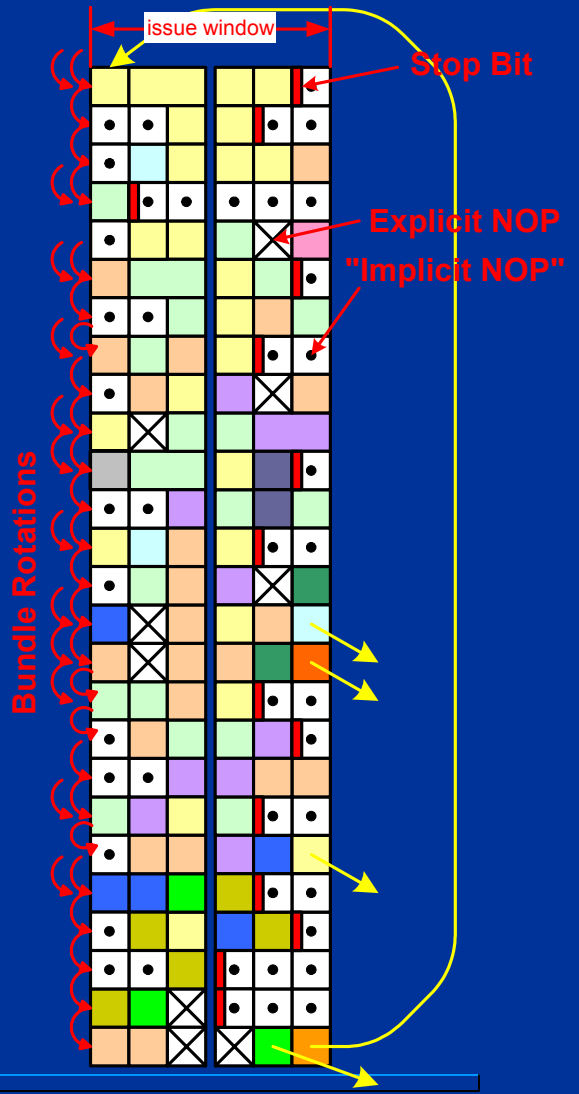
Ex: 175.vpr *route_net()* (SPECcpu2000)



- Base optimizations: function inlining, hyperblock formation, and loop unrolling
- Aggressive redundant load and operation elimination (red) [saves 18 cycles of 65]
- Advanced pointer analysis could disambiguate loads and stores (green) [save 5 additional cycles]

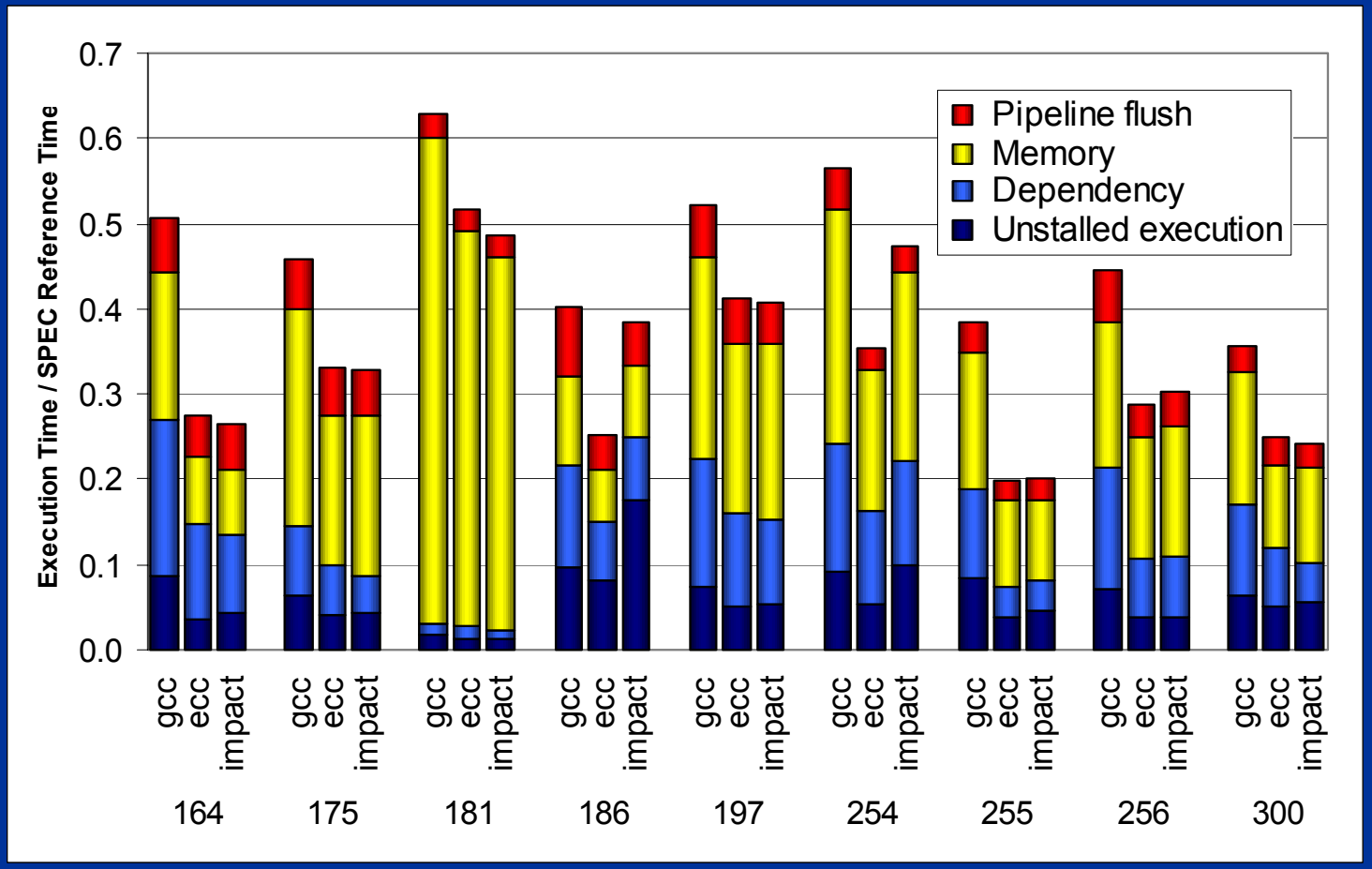
IPC and EPIC Instruction Issue





- Available ILP varies widely
- Bundling moderates between code size and efficient parallel issue

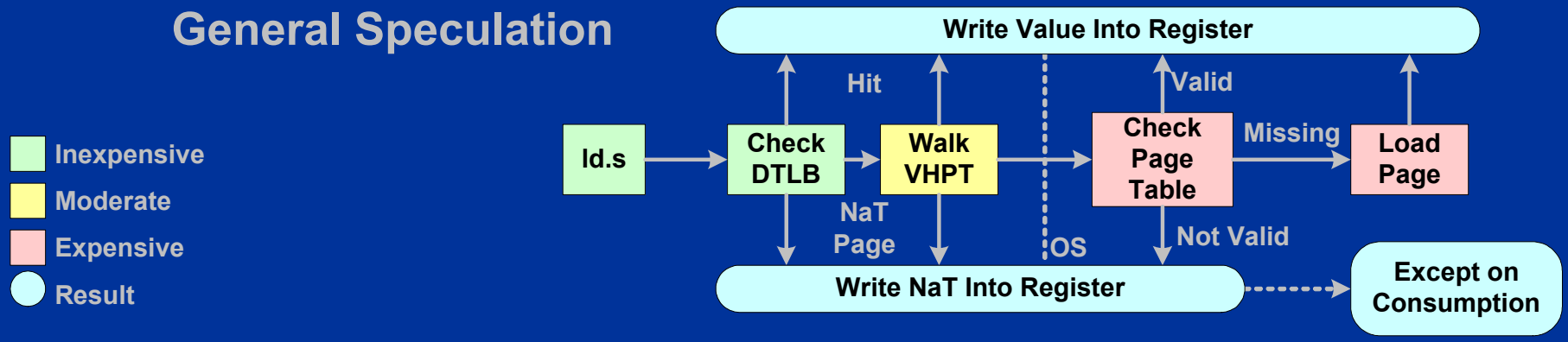
Cycle Accounting Breakdown



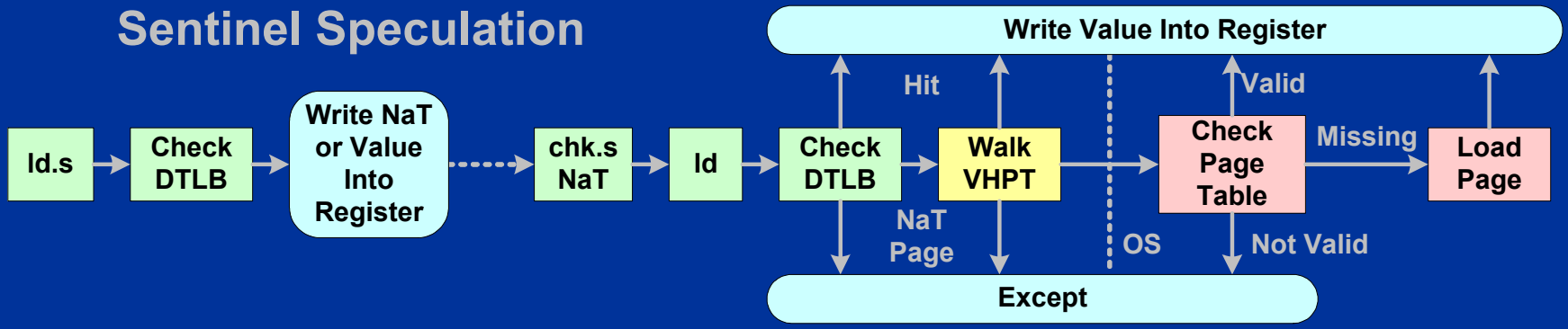
Measured machine execution cycles using performance monitoring counters

Approaches to Control Speculation

General Speculation

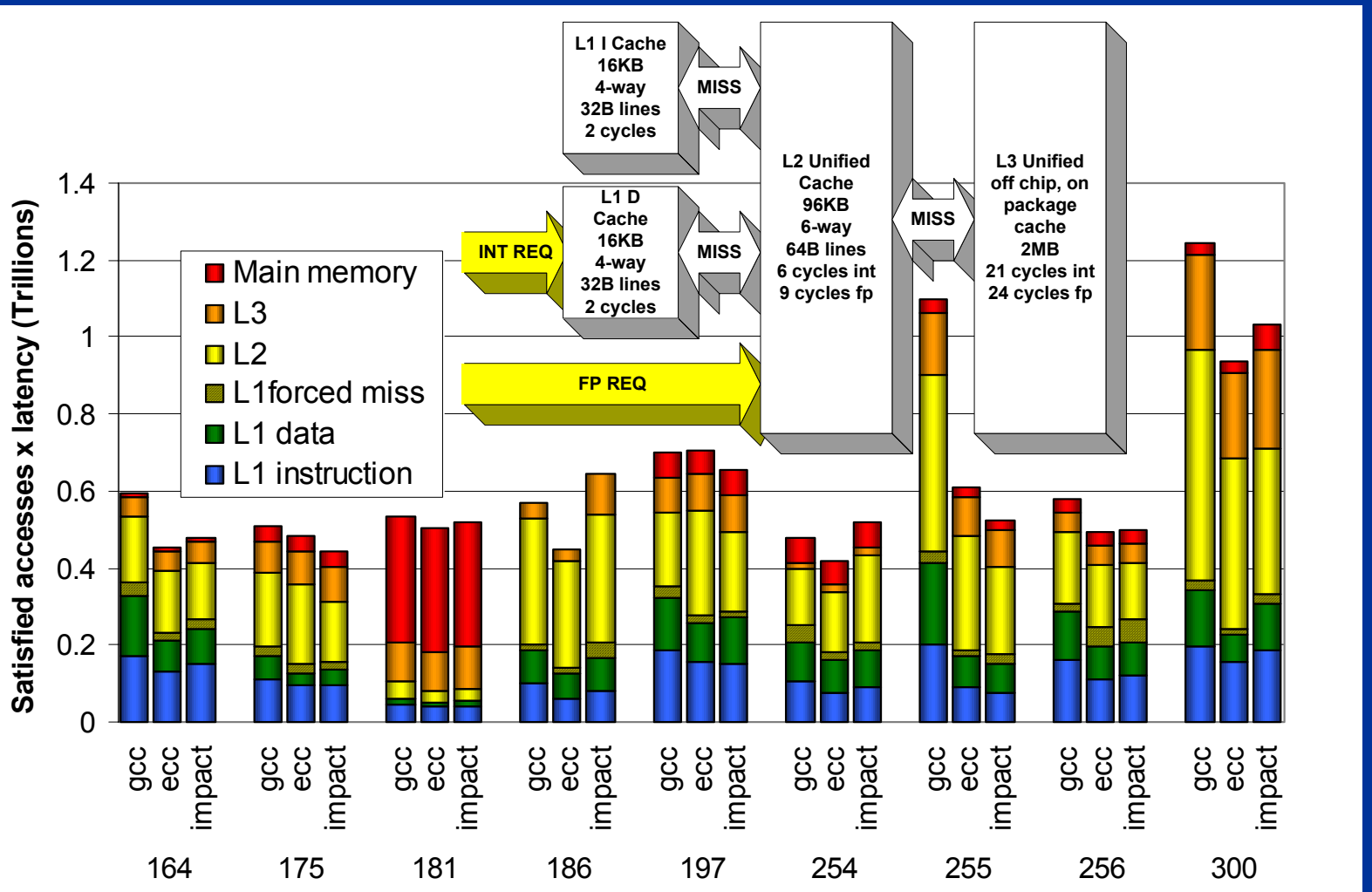


Sentinel Speculation



- IMPACT uses general speculation model
  - Recoverable exceptions handled immediately
  - No recovery code required

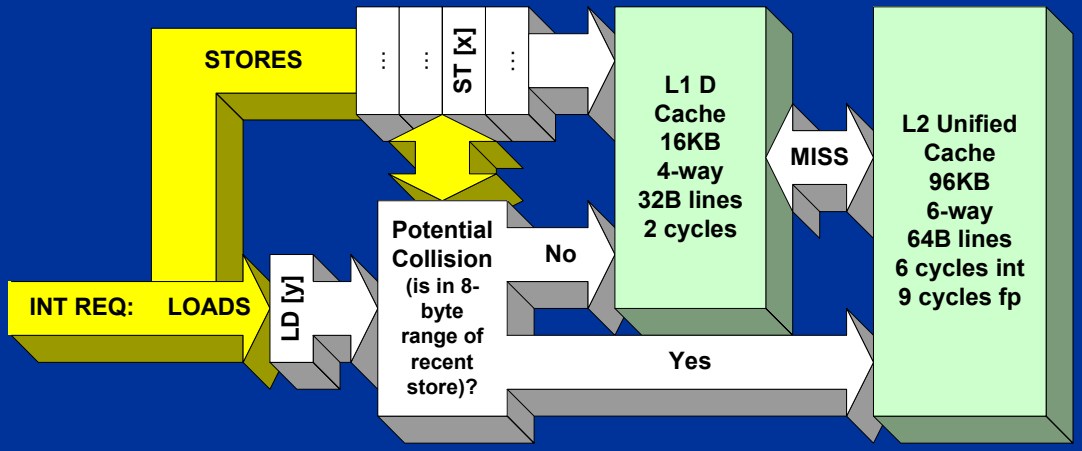
Performance of Memory Hierarchy



IMPACT ADVANCED COMPILER TECHNOLOGY

Exposed Memory Access Realities

- Loss of "ideal memory" an unavoidable eventuality
- Compiler cannot assume store-load forwards are free
- Penalties for forwarding at a 64-bit granularity
- Pointer and array dependence analysis crucial
- More tools for expressing cache control, but sophisticated analysis is required



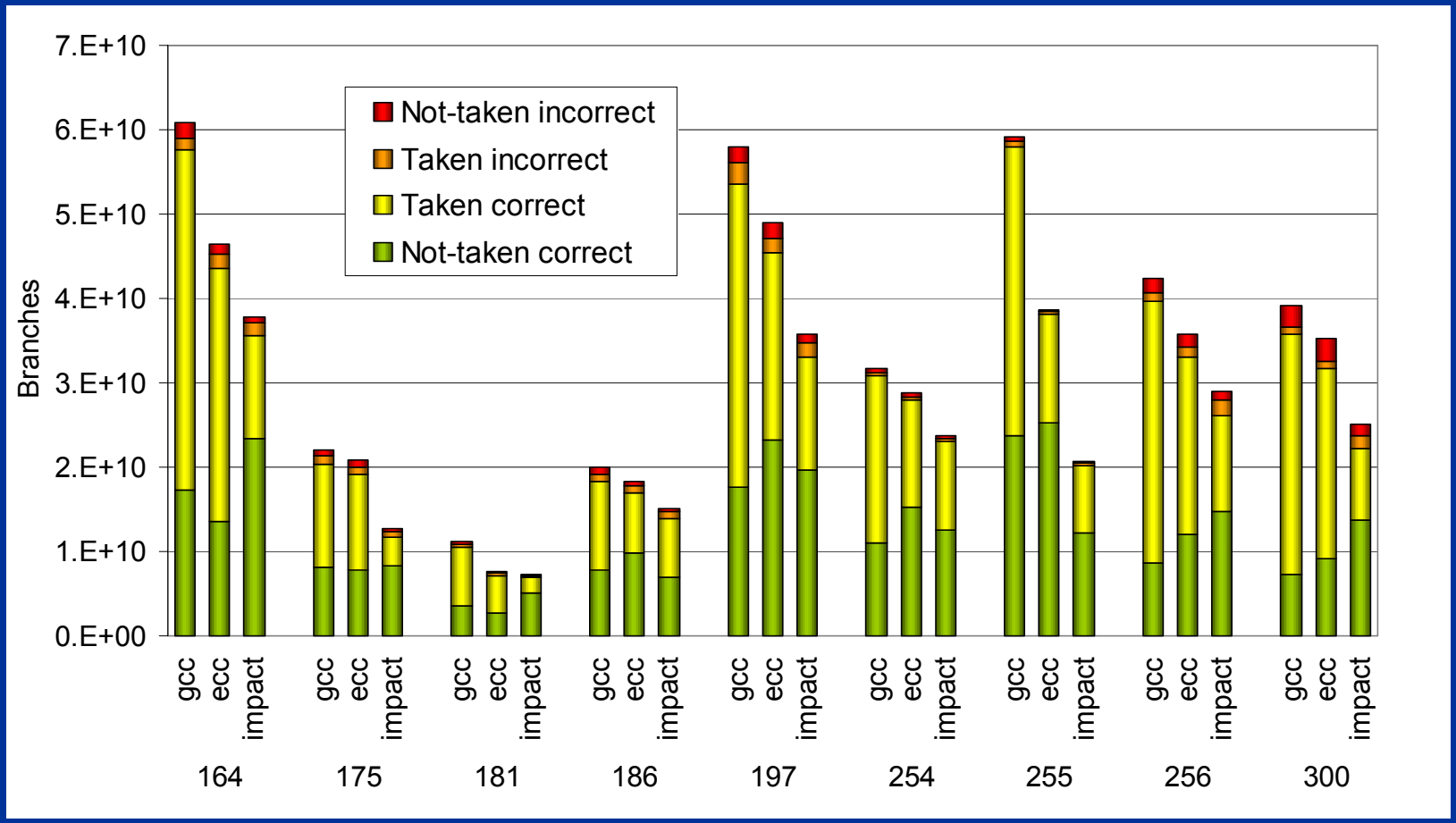
Original

```
cmp p1 =
If p1: st [r2] = x
Id y = [r2]
```

Partial Store Forwarding

```
cmp p1 =
Id y = [r2]
If p1: st [r2] = x
If p1:  y = x
```

Comparison of Branch Behavior



Reflections and Projections

- Compiler integral part of performance
  - Itanium benefits substantially from carefully targeted optimization
  - Brand new algorithms will be deployed in the EPIC compilers
  - Existing software development models must be supported / augmented
- Today, a mix of mature and immature compiler technologies
  - First Itanium machines and compilers are competitive at par frequency
  - High performance on particular benchmarks
  - Targeted optimization of memory access required for performance growth
  - Performance monitoring hardware provides guidance, assessment
- Evolution of EPIC architectures
  - Revisions of architectures are likely as compilers mature and microarchitectures become more exposed
  - Opportunities extend into load-time and run-time optimizations
- Significant performance headroom in product compilers

Acknowledgements

- Former IMPACT members
  - David August, Ben-Chung Cheng, Daniel Connors, Kevin Crozier, Brian Deitrich, John Gyllenhaal, Richard Hank, Teresa Johnson, Dan Lavery, Scott Mahlke, Le-Chun Wu
- Intel Itanium Team
  - Carole Dulong, John Crawford, Dan Lavery, Steve Skedzielewski, Jim Pierce
- HP Itanium Team
  - Richard Holman, Vatsa Santhanam, Carol Thompson, Richard Hank
- HP Labs PD Team
  - Bob Rau, Mike Schlansker, Vinod Kathail, Scott Mahlke
- HP Labs Linux Team
  - Brian Lynn, David Mosberger, Hans Boehm, Stephane Eranian
- HP Philanthropy - 9 i2000 Itanium machines, expedited
  - Karen Fontana, Tony Napolitan, Ralph Hyver, Chris Hsiung, Rob Bouzon
- Intel - 2 alpha/beta Itaniums
  - Carole Dulong, Richard Wirt