



rePLay: A Hardware Framework for Dynamic Program Optimization

Sanjay J. Patel

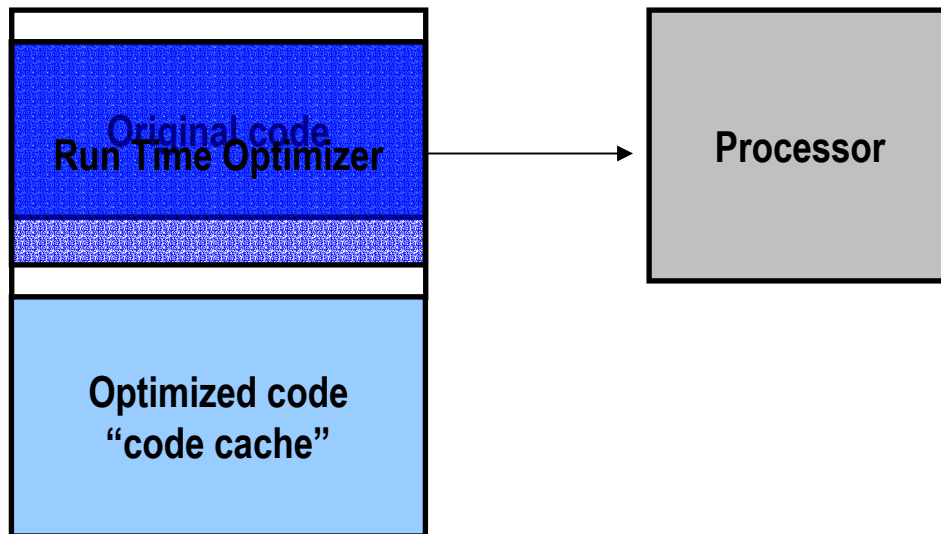
Center for Reliable and High-Performance Computing
Dept of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign



Outline

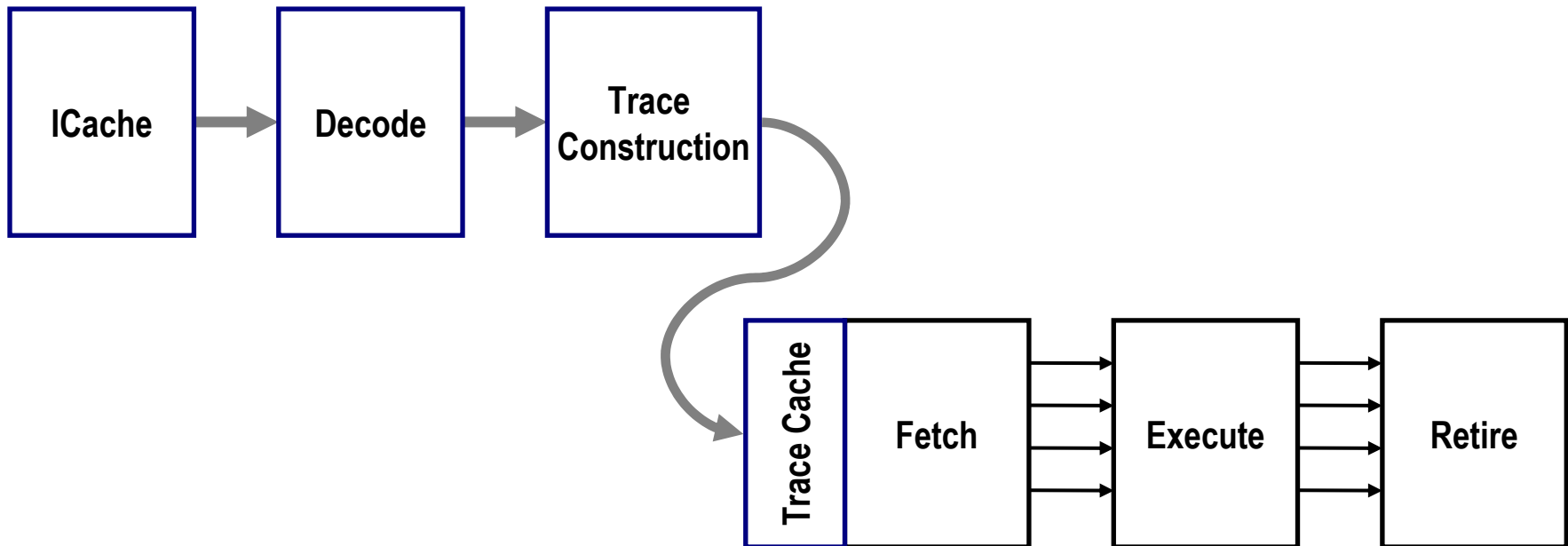
- rePLay: **hardware-based dynamic optimization**
- Limits of Dynamic Optimization: **how far can we push this?**
- Hardware Implementation: **can we actually build it?**

A quick background on dynamic optimization

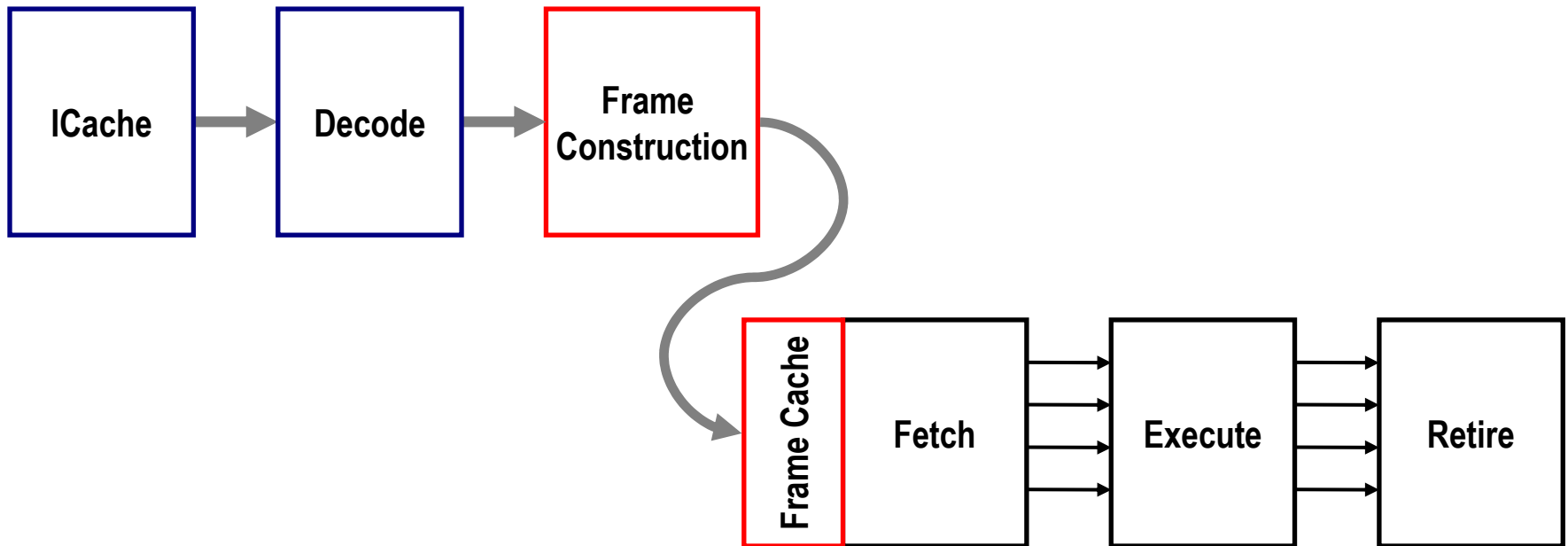


- Run-time optimizer monitors execution
- Run-time info boosts compiler optimization
 - Control information
 - Data values
 - Memory behavior
- Significant amount of recent development
 - Dynamo, Transmeta, ...

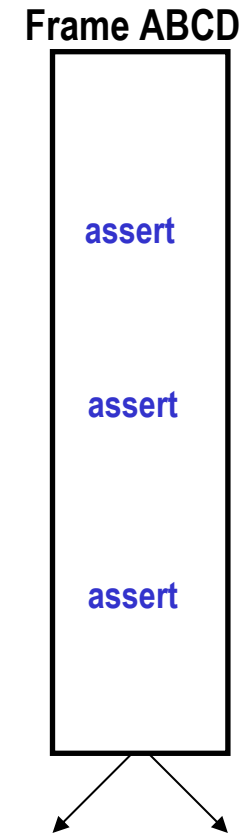
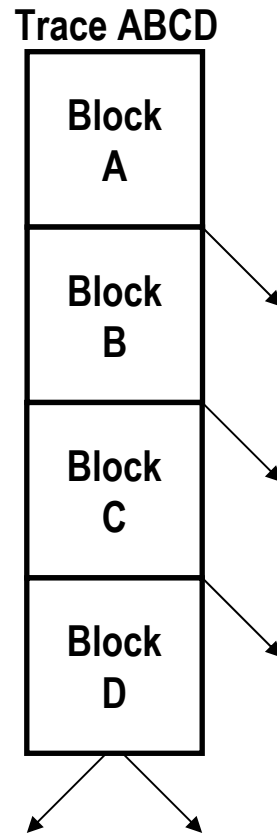
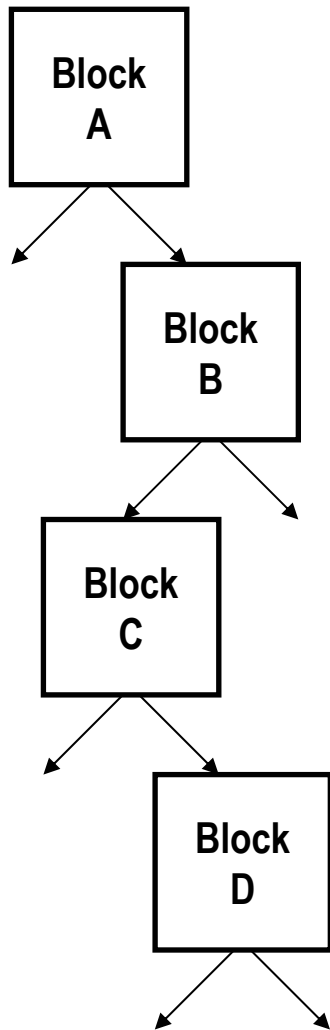
A High-Performance x86 Pipeline



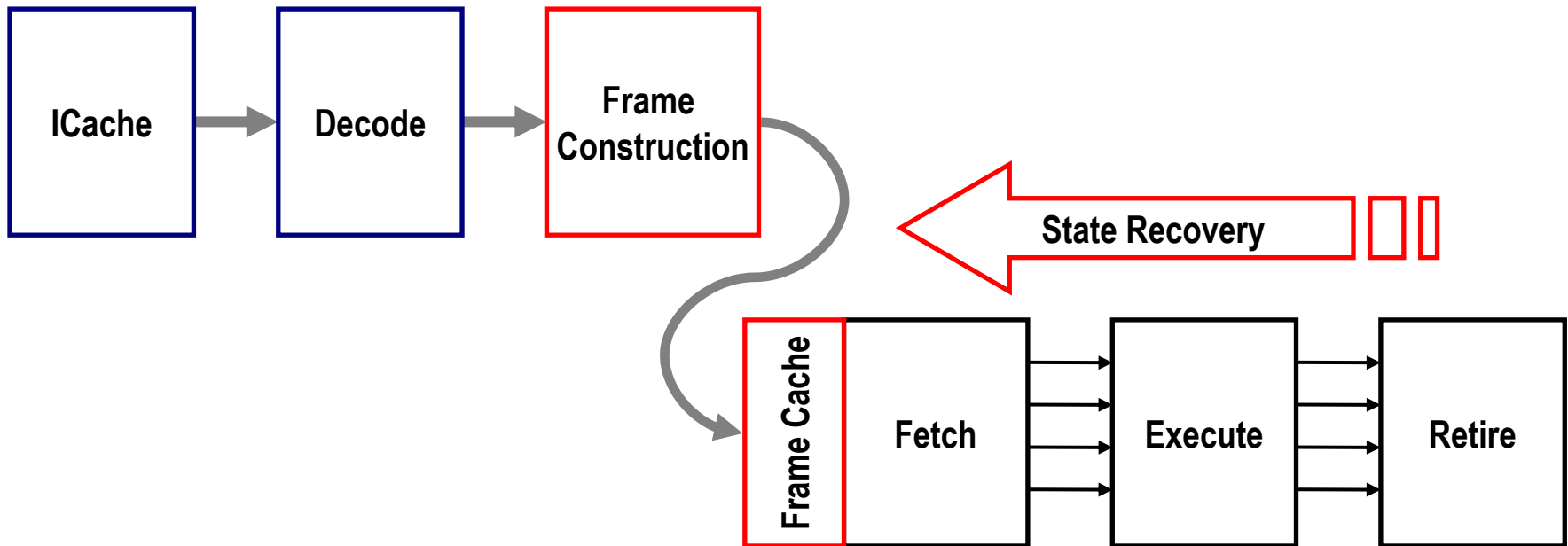
Frame Processing



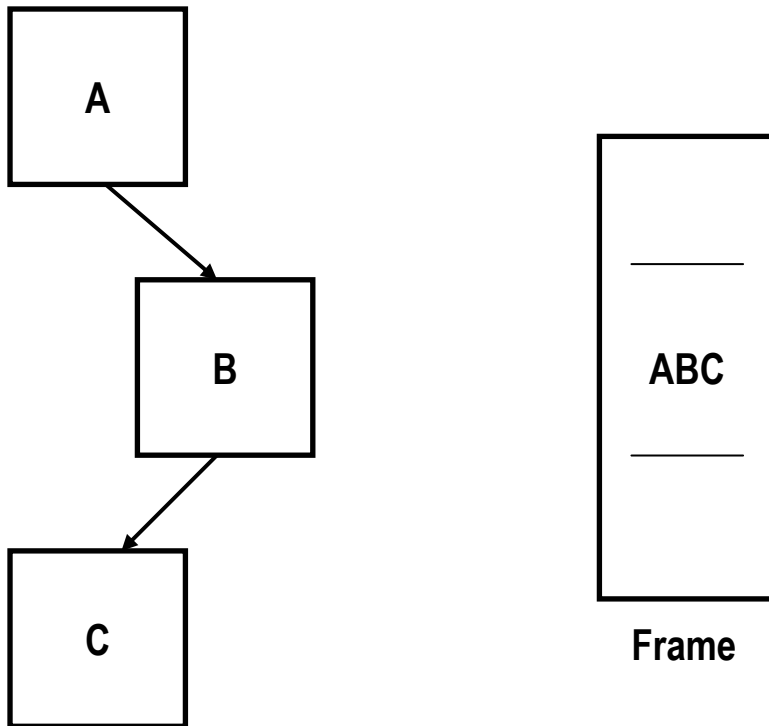
Frames: Atomic Traces



Frame Processing



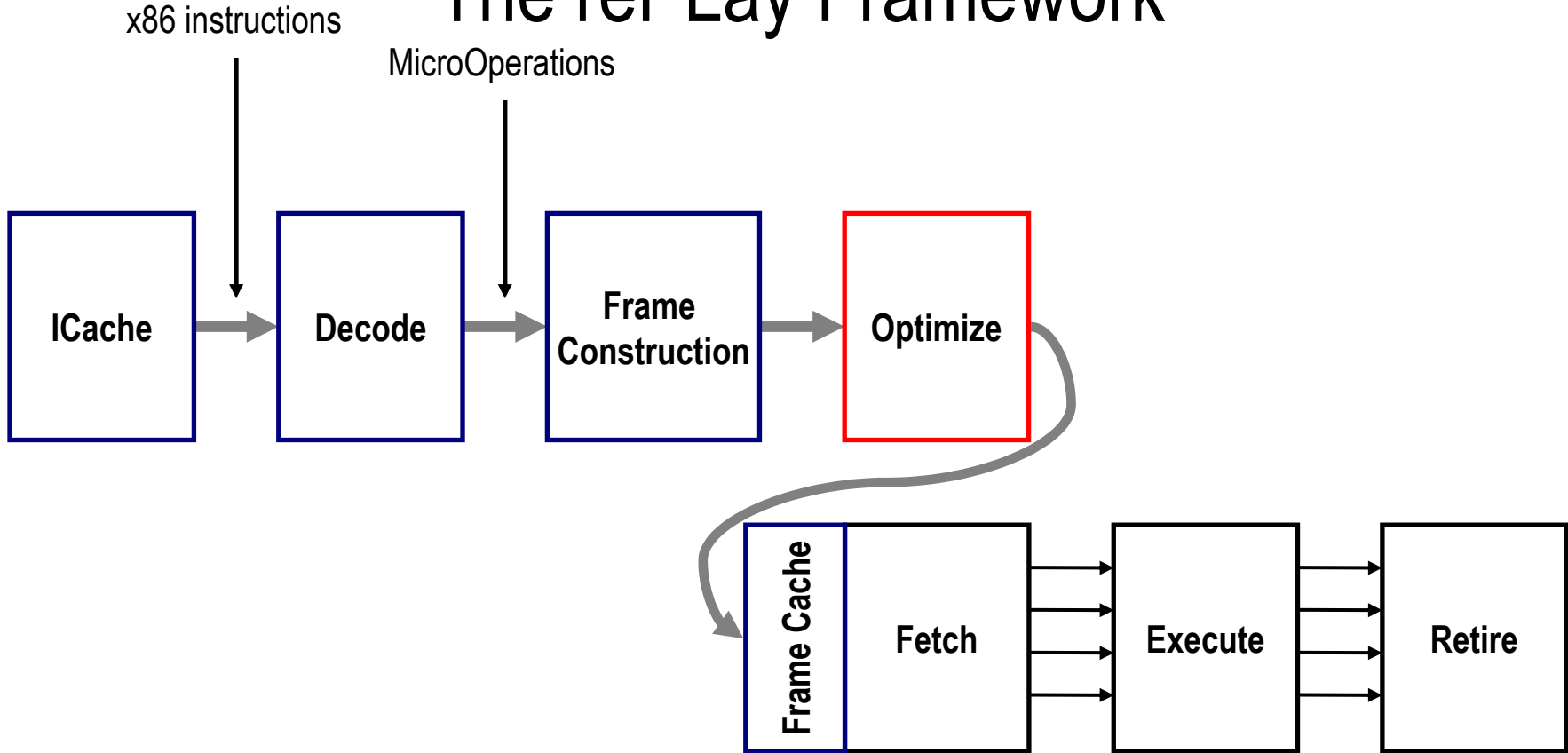
Frame Construction



- Frames are **atomic**, which helps with optimization
- ~80% of branches are converted into assertions
- Ave frame size: 60-100 insts
- High coverage, very low assertion rate

S. J. Patel, T. Tung, S. Bose, and M. Crum, "Increasing the Size of Atomic Instruction Blocks using Control Flow Assertions," 33rd Annual Symposium on Microarchitecture, December 2000.

Speculative Optimization with The rePLAY Framework



B. Slechta et al., “Dynamic Optimization of Micro-instructions”, 9th Annual Symposium on High-Performance Computer Architecture, February 2003.

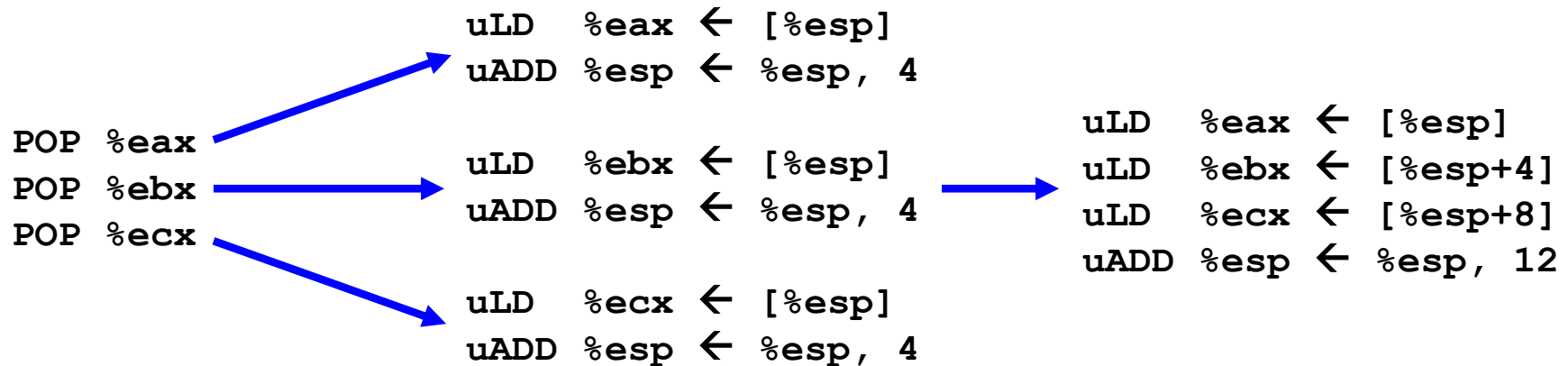
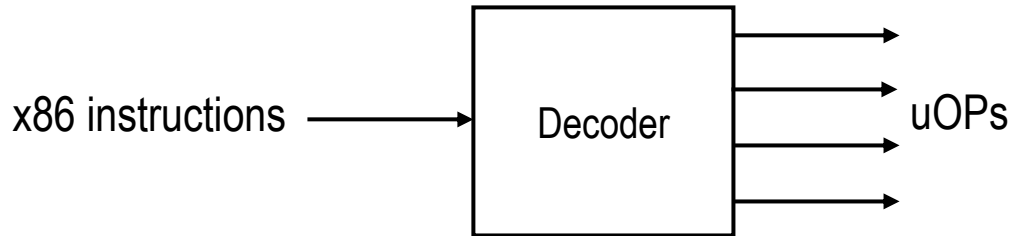
B. Fahs et al., “Performance Characterization of a Microarchitectural Framework for Dynamic Optimization,” 34th Annual Symposium on Microarchitecture, December 2001.

Advanced Computing Systems Group
Sanjay J. Patel

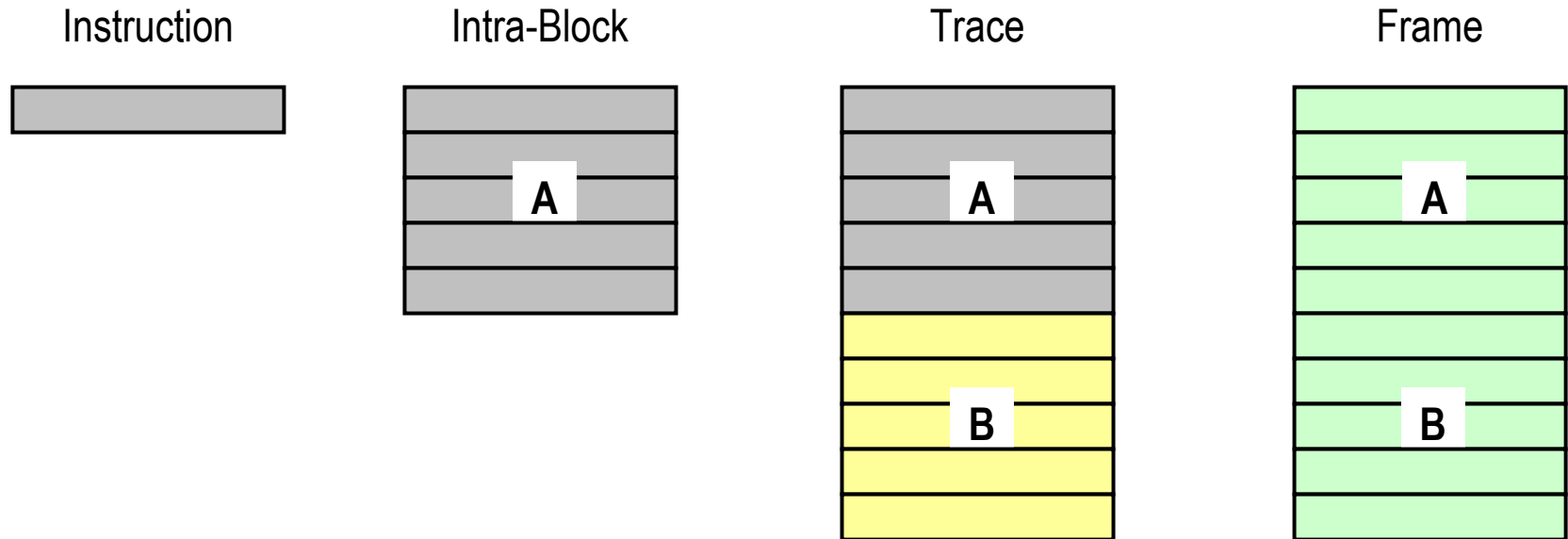


ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Decoding x86 Instructions



Micro-Operation Optimization Requires Raising Architectural Boundaries



Simple Optimizations

- Dead Code Removal
- Constant Propagation
- Reassociation
- Common Subexpression Elimination
- Store Forwarding
- List Scheduling
- Strength Reduction

Basic Block Optimization

x86 instructions

```
PUSH EBP
PUSH EBX
MOV EBX, [ESP+10H]
MOV EDX, ECX
OR EDX, EBX
JZ 15H <Block2>
```

Block2 :

```
POP EBX
POP EBP
```

micro-operations

```
01 [ESP-4] <- EBP
02 ESP <- ESP-4
03 [ESP-4] <- EBX
04 ESP <- ESP-4
05 EBX <- [ESP+16]
06 EDX <- ECX
07 EDX, flags <- EDX|EBX
08 if (flags == 0)
    jump <Block2>
```

```
09 EBX <- [ESP]
10 ESP <- ESP+4
11 EBP <- [ESP]
12 ESP <- ESP+4
```

result of optimization

```
01 [ESP-4] <- EBP
02 ESP <- ESP-4
03 [ESP-8] <- EBX
04 ESP <- ESP-8
05 EBX <- [ESP+16]
06 EDX <- ECX
07 EDX, flags <- ECX|EBX
08 if (flags == 0)
    jump <Block2>
```

```
09 EBX <- [ESP]
10 ESP <- ESP+4
11 EBP <- [ESP+4]
12 ESP <- ESP+8
```

Trace Optimization

x86 instructions	micro-operations	result of optimization
PUSH EBP	01 [ESP-4] <- EBP	01 [ESP-4] <- EBP
PUSH EBX	03 [ESP-8] <- EBX	03 [ESP-8] <- EBX
	04 ESP <- ESP-8	04 ESP <- ESP-8
MOV EBX, [ESP+10H]	05 EBX <- [ESP+16]	05 EBX <- [ESP+16]
MOV EDX, ECX		
OR EDX, EBX	07 EDX, flags <- ECX EBX	07 EDX, flags <- ECX EBX
JZ 15H <Block2>	08 if (flags == 0) jump <Block2>	08 if (flags == 0) jump <Block2>
Block2:		
POP EBX	09 EBX <- [ESP]	09 EBX <- [ESP]
POP EBP	11 EBP <- [ESP+4]	11 EBP <- [ESP+4]
	12 ESP <- ESP+8	12 ESP <- ESP+8

Atomic Region Optimization

x86 instructions	micro-operations	result of optimization
PUSH EBP	01 [ESP-4] <- EBP	01 [ESP-4] <- EBP
PUSH EBX	03 [ESP-8] <- EBX	03 [ESP-8] <- EBX
	04 ESP <- ESP-8	04 ESP <- ESP-8
MOV EBX, [ESP+10H]	05 EBX <- [ESP+16]	05 TMP <- [ESP+16]
MOV EDX, ECX		
OR EDX, EBX	07 EDX, flags <- ECX EBX	07 EDX, flags <- ECX TMP
JZ 15H <Block2>	08 if (flags == 0) jump <Block2>	08 ASSERT (flags == 0)
Block2:		
POP EBX	09 EBX <- [ESP]	09 EBX <- [ESP]
POP EBP	12 ESP <- ESP+8	12 ESP <- ESP+8

Processor Configurations

- ICache (**IC**)
 - Simple Trace Cache (**TC**)
 - rePLay without Optimizations (**RP**)
 - rePLay with Optimizations (**RPO**)
-

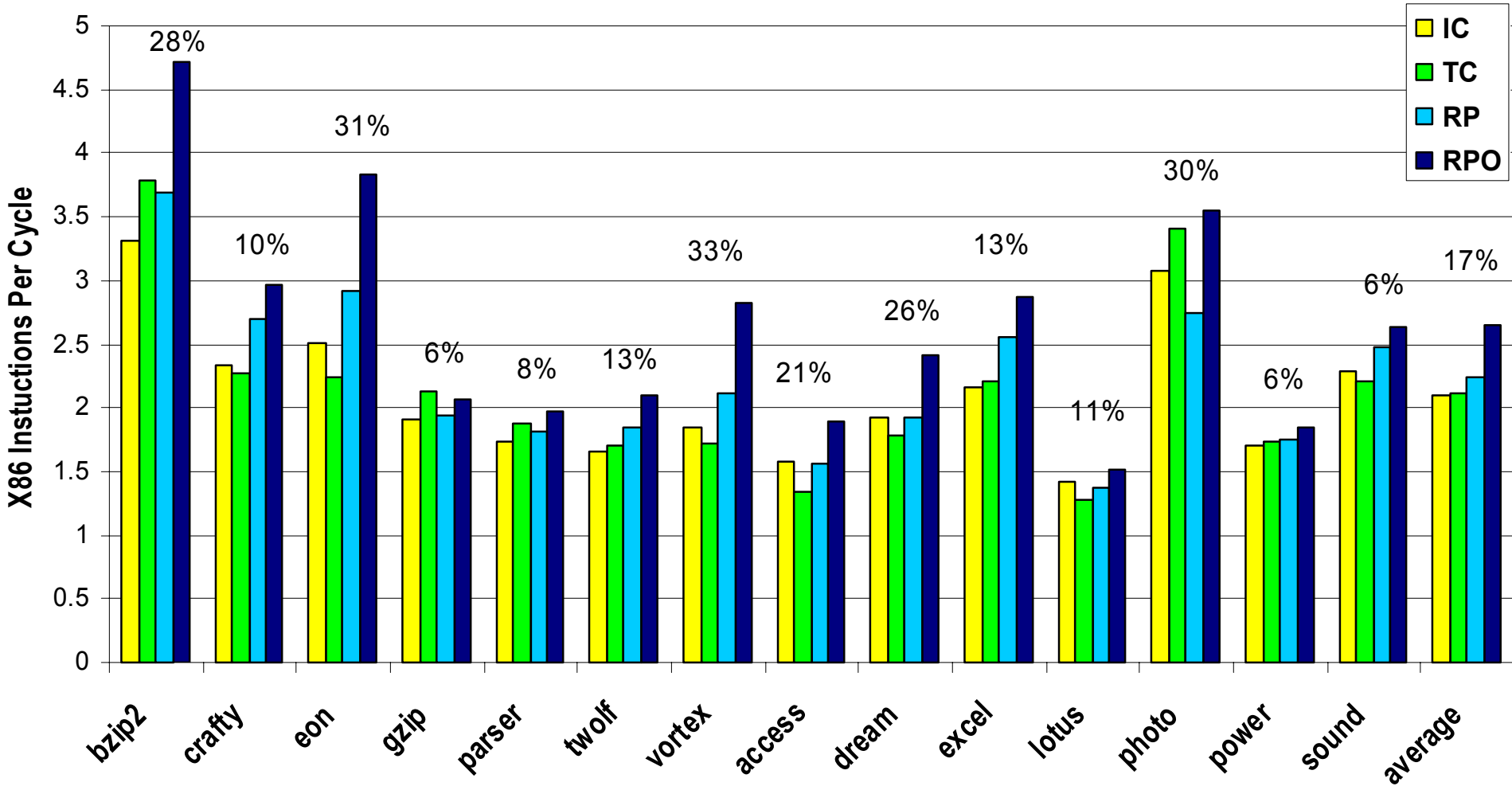
Fetch: 8-wide, 64KB of cache space for insts.

Core: 8-wide OOO, 64KB DCache, 1 exe cluster
15 cycles (min) for branch resolution

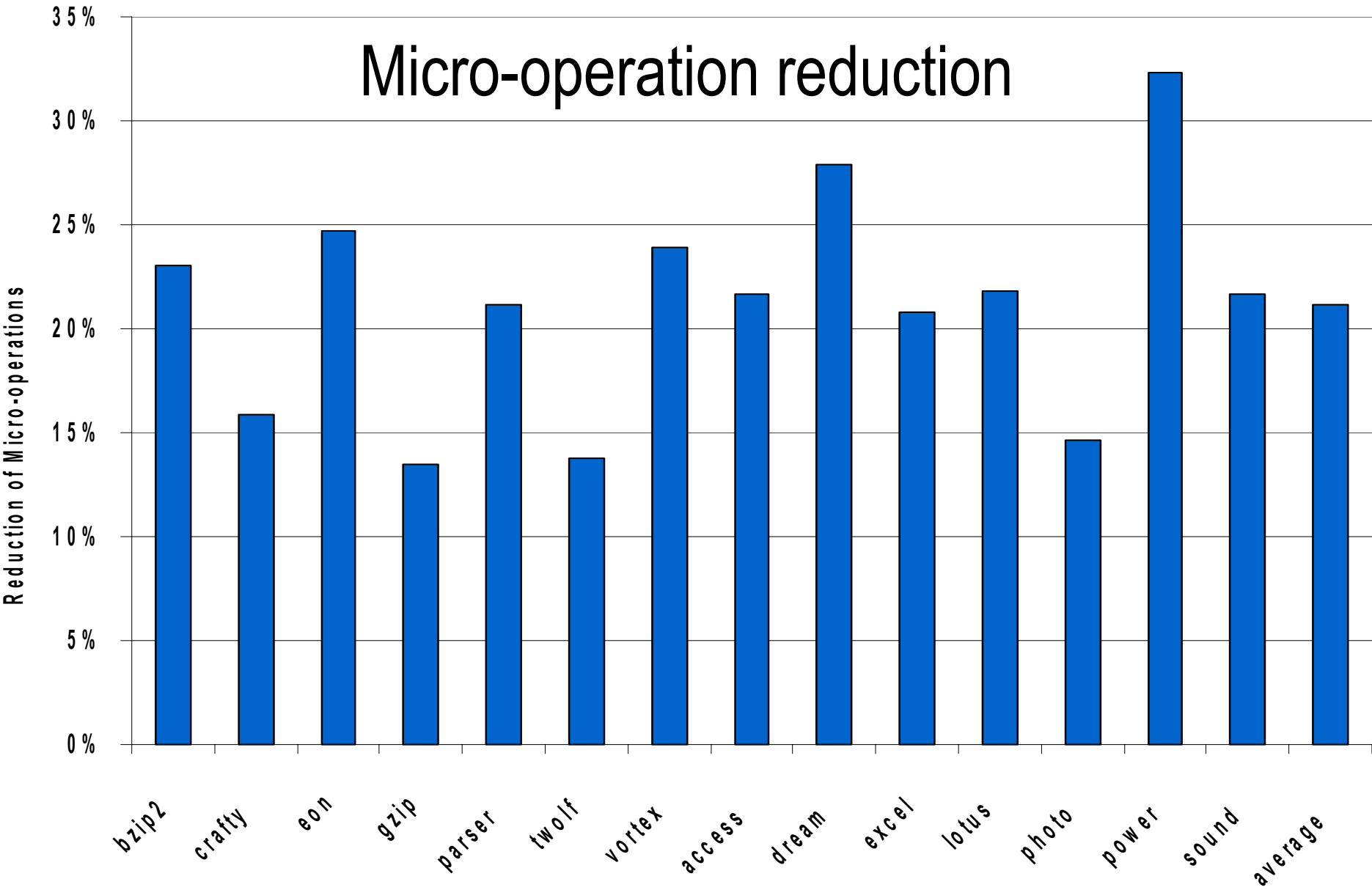
Optimizer: 10 cycles/inst, non-pipelined, 3 frame buffer

x86 Workload traces provided by AMD

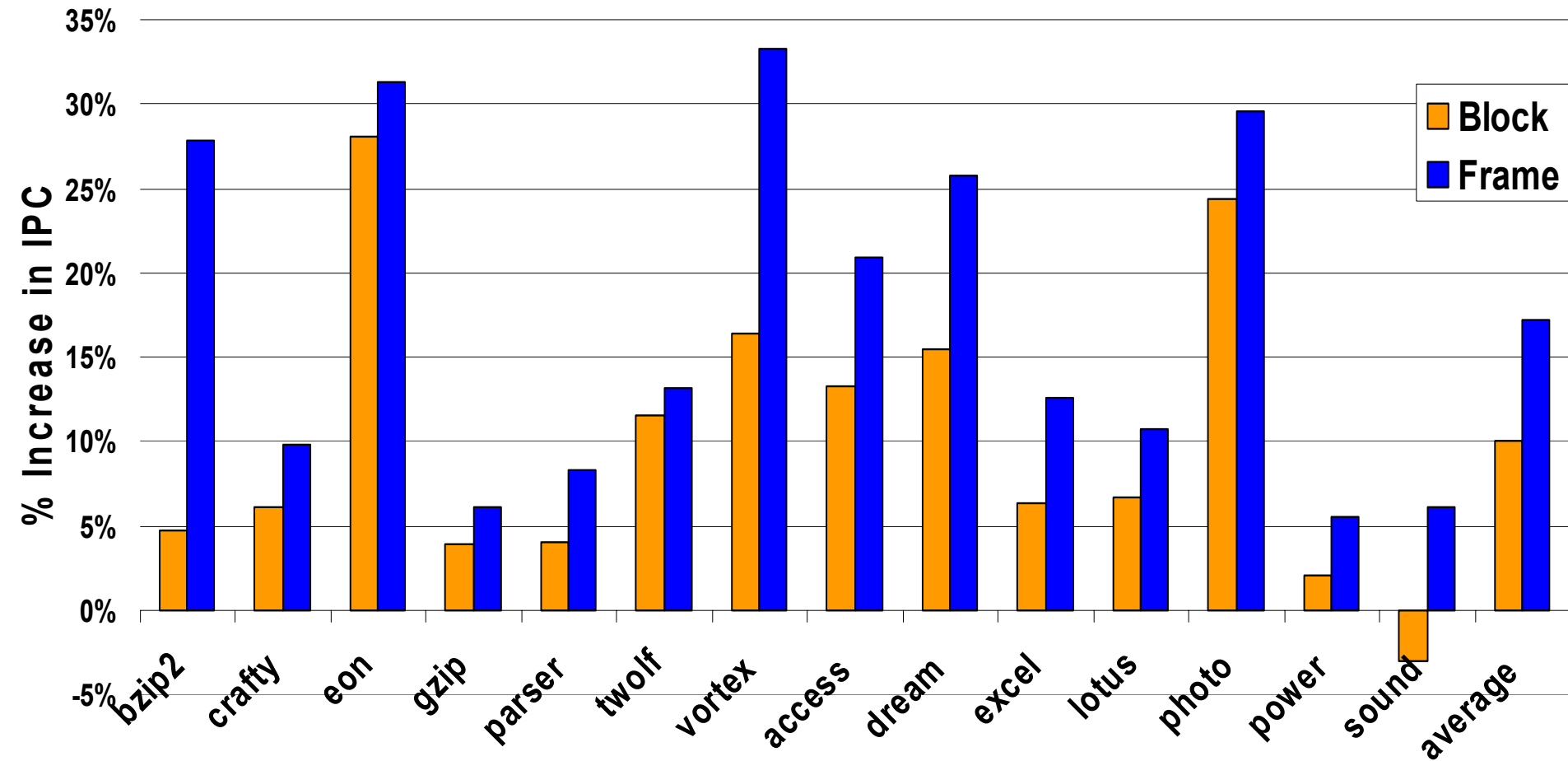
Increase in ILP due to MicroOp Optimization



Micro-operation reduction



Optimization of Blocks vs. Frames

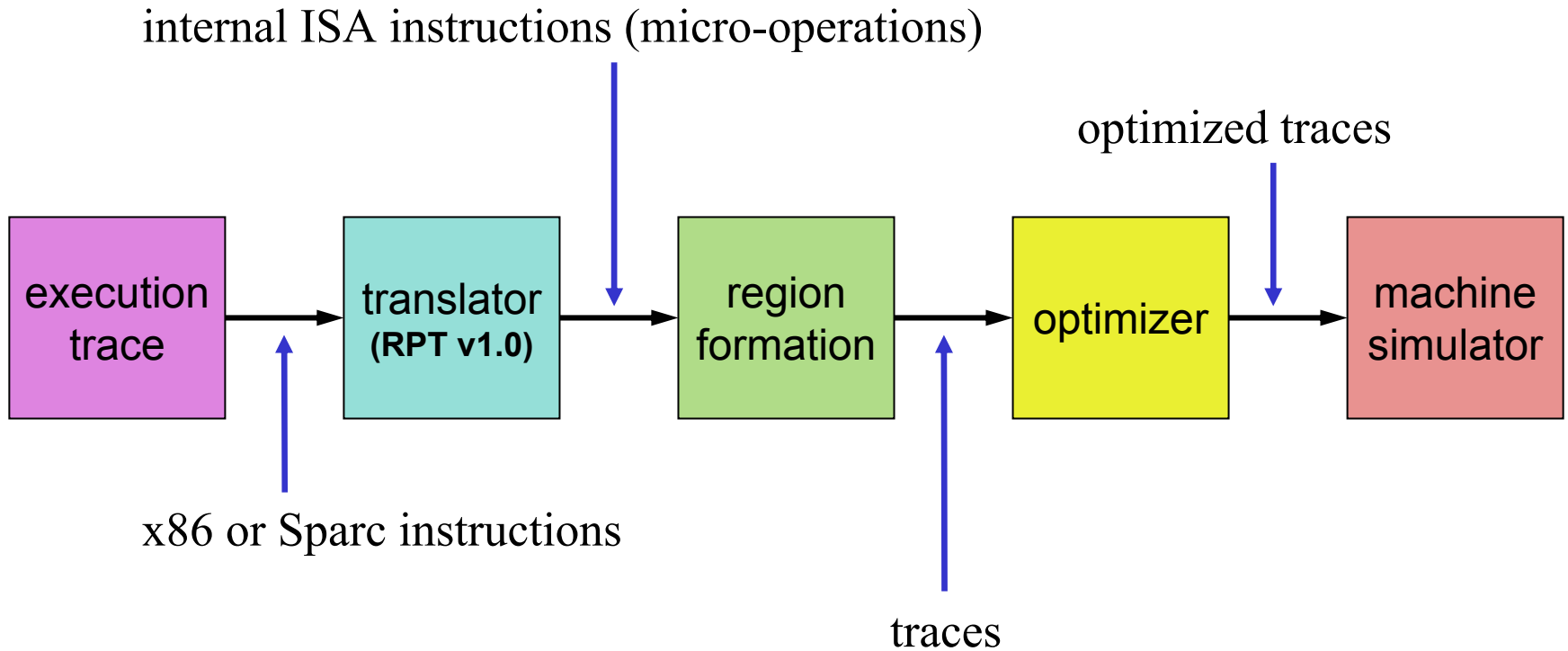


Other types of optimizations

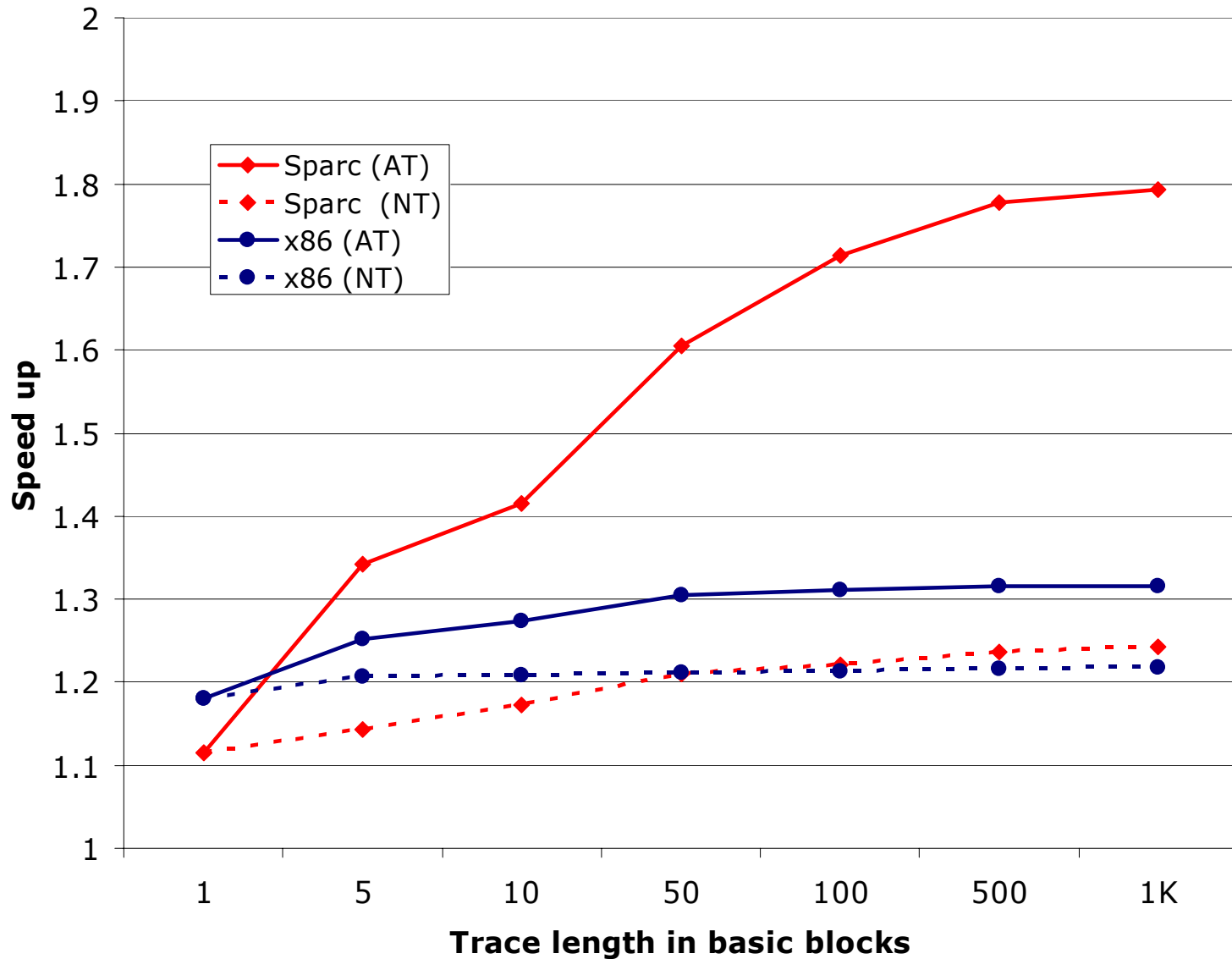
- SIMD-ification: **Parrot Project at Intel**
- Power reduction: **Parrot Project at Intel**
- Value specialization: **using data assertions**
- Idiomization: **processor-specific instructions**
- Soft-error protection: **assertion-based recovery**

The Potential of Dynamic Optimization

Experimental Setup



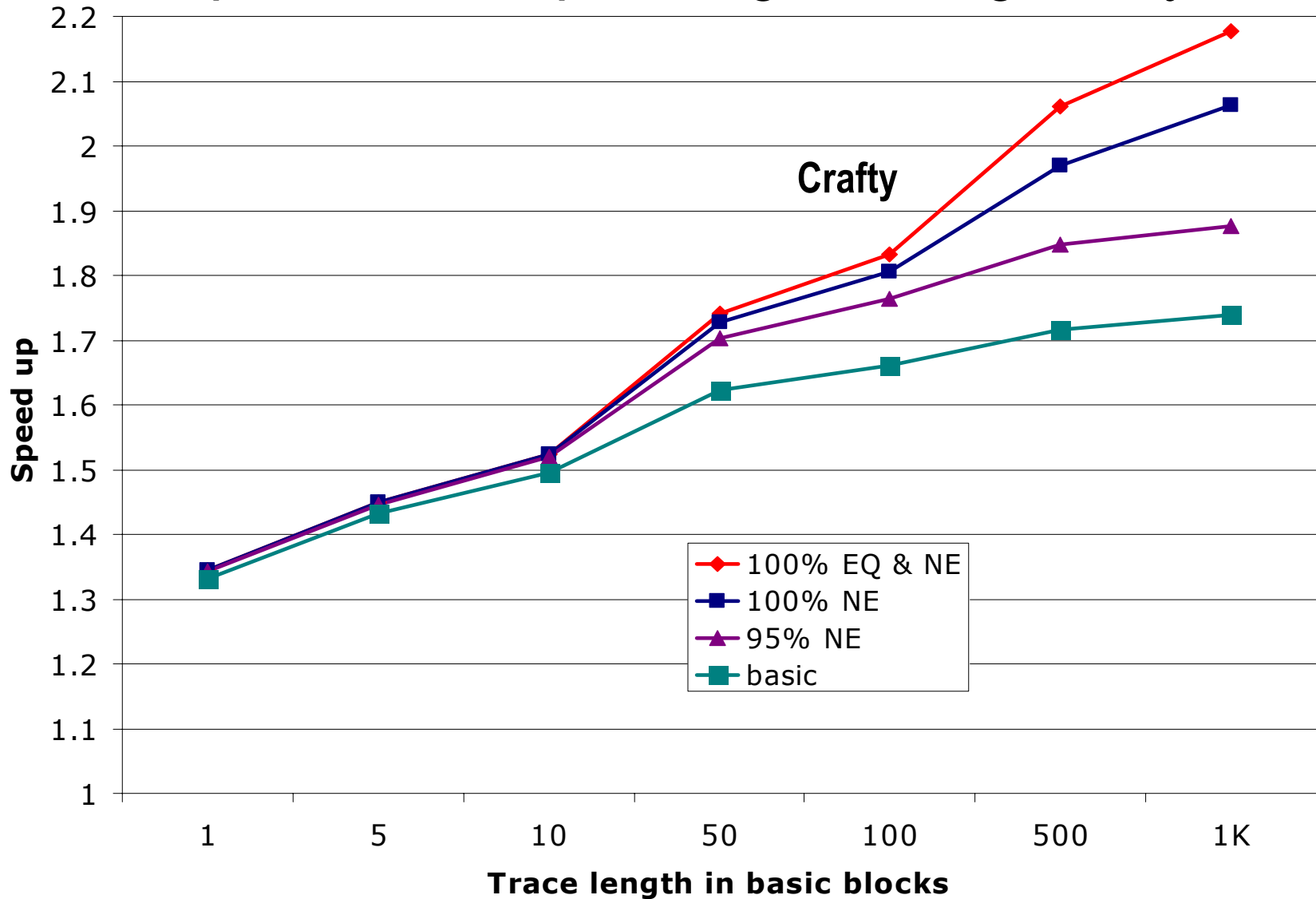
x86/SPARC Long Trace Performance



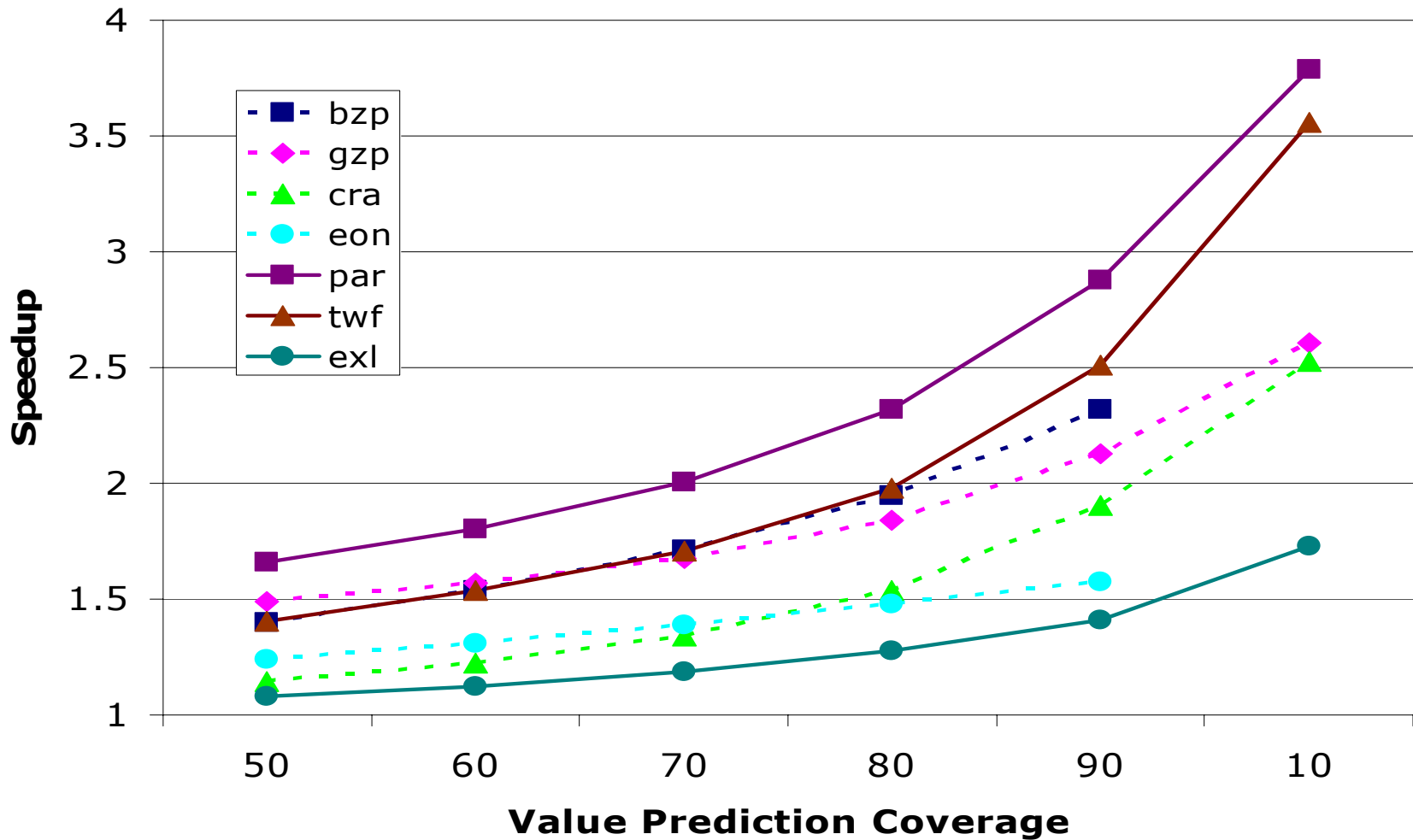
What can we draw from this data

- Dynamic optimization potential between 20% and 90%
- Atomic traces provide significantly better potential
- Non-atomic trace potential independent of trace length
- Sparc potential much higher than x86 potential

Impact of Incorporating Aliasing Analysis



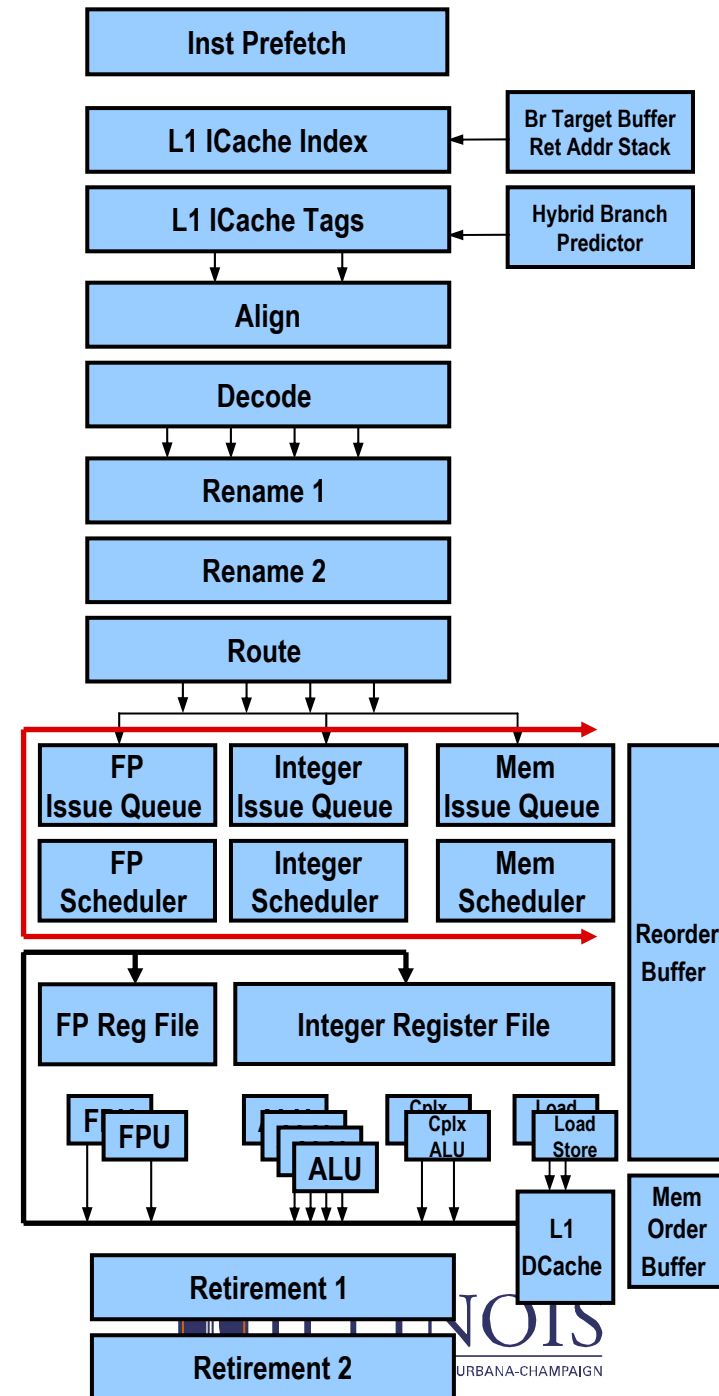
Value Specialization of Traces



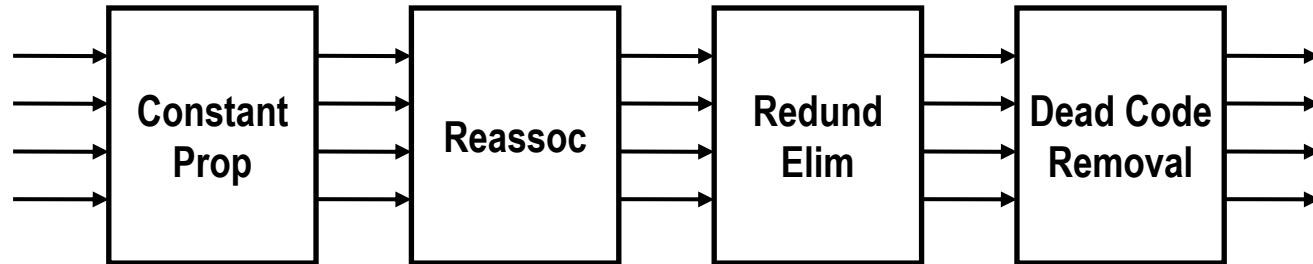
Reduction to hardware [Current Project]

Verilog Processor Model

- Modern pipeline:
 - 4-wide, 15-stage dynamically-scheduled processor, based on the Pentium 4/Athlon
 - Executes Alpha ISA
- Heavy amount of speculation:
 - Complex branch prediction, I+DCache way prediction, spec wakeup, mem dep predictor, inst and data prefetching
- Latch-level fidelity:
 - Approx 50K bits throughout the pipeline.



A Pipelined Table-based HW Optimizer



- This optimizer consists of 3-5 pipeline stages in after the decoder
- Register file-like tables store information about properties of an instruction's source registers
- Dead code removal is a little problematic

I'm done

Cast of Characters

- Little of this would be possible without :
 - David Crowe, Todd Ehrhart, Brian Fahs, Joe Grzywacz, Aqeel Mahesri, Greg Muthler, Justin Quek, Todd Rafacz, Galen Rasche, Kapil Sachdeva, Francesco Spadini, Nick Wang, and ACS students of the past.
 - Profs. Matthew Frank, Steve Lumetta, and Wen-mei Hwu, all at UIUC.
 - C2S2 MARCO Center, NSF, AMD, Intel, IBM, and Sun