Advanced Microarchitecture

Yale N. Patt
The University of Texas at Austin

ACACES 2006
L'Aquila, Italia
July 24-28, 2006

About the Instructor

Yale Patt is Professor of Electrical and Computer Engineering, Professor of Computer Sciences, and the Ernest Cockrell, Jr. Centennial Chair in Engineering at The University of Texas at Austin. He directs the research of 11 PhD students in high performance computer architecture and implementation, and enjoys teaching both large undergraduate classes and small advanced graduate seminars. He has, for more than 35 years, combined an active research program with extensive consulting in industry and a strong commitment to teaching. The research he has conducted with his students has had major impact on the microprocessor industry. HPS (the integration of wide-issue, speculative, out-of-order execution, and in-order retirement), the two-level branch predictor, and SSMT (more commonly called helper threads) are three examples. Dr. Patt has received many awards for his research and teaching, including the IEEE Emmanuel R. Piore Medal in 1995, the IEEE/ACM Eckert-Mauchly Award in 1996, the IEEE Wallace W. McDowell Award in 1999, the ACM Karl V. Karlstrom Outstanding Educator Award, in 2000, and most recently, the IEEE Charles Babbage Award in 2005. His vital concern for how we introduce computing to computer science and engineering majors has led to "Introduction to Computing Systems, from Bits and Gates to C and Beyond," co-authored with Professor Sanjay Patel of The University of Illinois.

Yale Patt earned his BS at Northeastern and MS and PhD at Stanford, all in electrical engineering. He is a Fellow of both the IEEE and of the ACM. For a more detailed bio, see his web site: http://www.ece.utexas.edu/~patt/

Five Lectures

Monday: Introduction and Focus

Tuesday: Microarchitecture Structures

Wednesday: Microarchitecture Structures (cont.)

Thursday: Alternate Approaches to Concurrency

Friday: Microarchitecture: Today and Tomorrow

Introduction and Focus

- ***The Fundamentals***

- ***Computer Architecture:***
  A Science of Tradeoffs

- ***Embedded Processors vs. General Purpose***

Fundamentals

* ***Overriding consideration:***

What does it cost?
What is the benefit?

* ***Global View***

--Global vs. Local transformations

* ***Microarchitecture view***

--The three ingredients to performance

* ***Physical view***

--Partitioning
--Power consumption
--Wire delay

Problem

Algorithm

Program

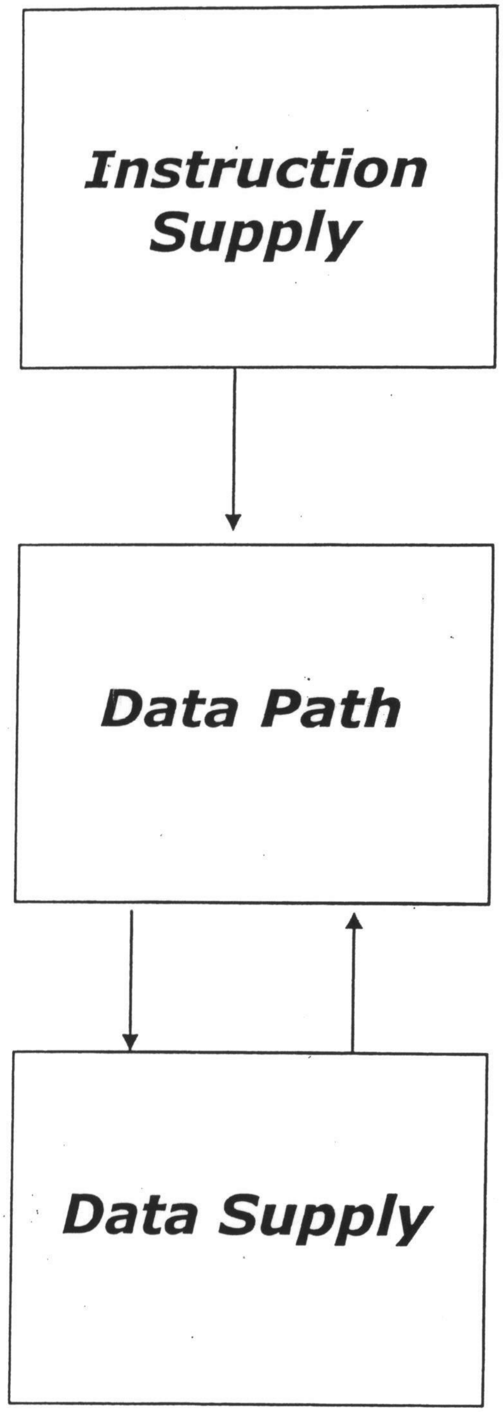
Instruction Set Architecture (ISA)

Microarchitecture

Circuits

Electrons

Microarchitecture
(The Requirement)



Instruction Supply

- *Perfect I-Cache*
- *No Packet Breaks*
- *100% Br. Pred.*

Data Path

- *Perfect Data Flow*

Irregular Parallelism

- *Enough Functional Units*
- *Perfect Intraconnect*

Data Supply

- *Infinite capacity*
- *Zero access time*

The Triangle

* Only the algorithm knows the programmer

  -- pragmas
  -- pointer chasing
  -- partition code, data

* Only the compiler knows the future (?? :-)

  -- Predication
  -- Prefetch/Poststore
  -- Block-structured ISA

* Only the hardware knows the past

  -- Branch directions
  -- Cache misses
  -- Functional unit latency

The Soul of Computer Architecture (A Science of Trade-offs)

* ***Choices/Trade-offs: The ISA***

-- Dynamic/Static Interface

-- Fixed-length, Uniform Decode vs…

-- Condition Codes vs…

-- Load/Store vs…

-- Help for Programmer vs help for Microarchitect (addressing modes, data types, unaligned access)

-- Hardware Interlocks vs…

-- VLIW vs…

-- 0, 1, 2, 3, address machines

-- Compatibility (e.g., delayed branch)

-- Precise exceptions vs…

Science of Trade-offs (continued)

* ***Choices/Trade-offs: The µarchitecture***

- ***-- CPI vs. t (or IPC vs frequency)***
- ***-- ASIC vs programmed control***
- ***-- in-order vs out-of-order***
- ***-- Speculative vs Stand around and wait***
- ***-- Superscalar***
- ***-- Use of Chip real estate (tomorrow: the Refrigerator!)***
- ***-- Pipeline depth***
- ***-- Partitioning (until when?)***

* ***Choices/Trade-offs: The System***

- ***-- MP granularity***
- ***-- Commodity vs Tailored part***
- ***-- What at compile time, what at runtime***

Other Fundamentals

∗ ***Microarchitecture***

- ***-- Critical Path Design***
- ***-- Bread and Butter Design***
- ***-- Balanced Design***

∗ ***New Technology Issues***

- ***-- Wire delay***
- ***-- Energy, Power, Temperature***
- ***-- High frequency → Proc/Mem imbalance***
- ***-- Soft errors***

Embedded Processors vs. General Purpose

- *What is different from what we've said?*
- *Easier to design wholistically*
  (limited purpose, special purpose)
- *Greater use of ASICS*
  (The x + superscalar syndrome)
- *Why VLIW*
- *Partitioning*
- *Memory latency*
- *Reconfigurability ??*

Finally,

∗ **_Role of the Architect_**

-- _Look Backward (Examine old code)_

-- _Look forward (Listen to the dreamers)_

-- _Look Up (Nature of the problems)_

-- _Look Down (Predict the future of technology)_

∗ **_Design Points_**

-- _Performance_

-- _Reliability_

-- _Availability_

-- _Cost_

-- _Power_

-- _Time to Market_

∗ **_Agents of Evolution_**

-- _Performance_

-- _Bottlenecks_

-- _Good Fortune_

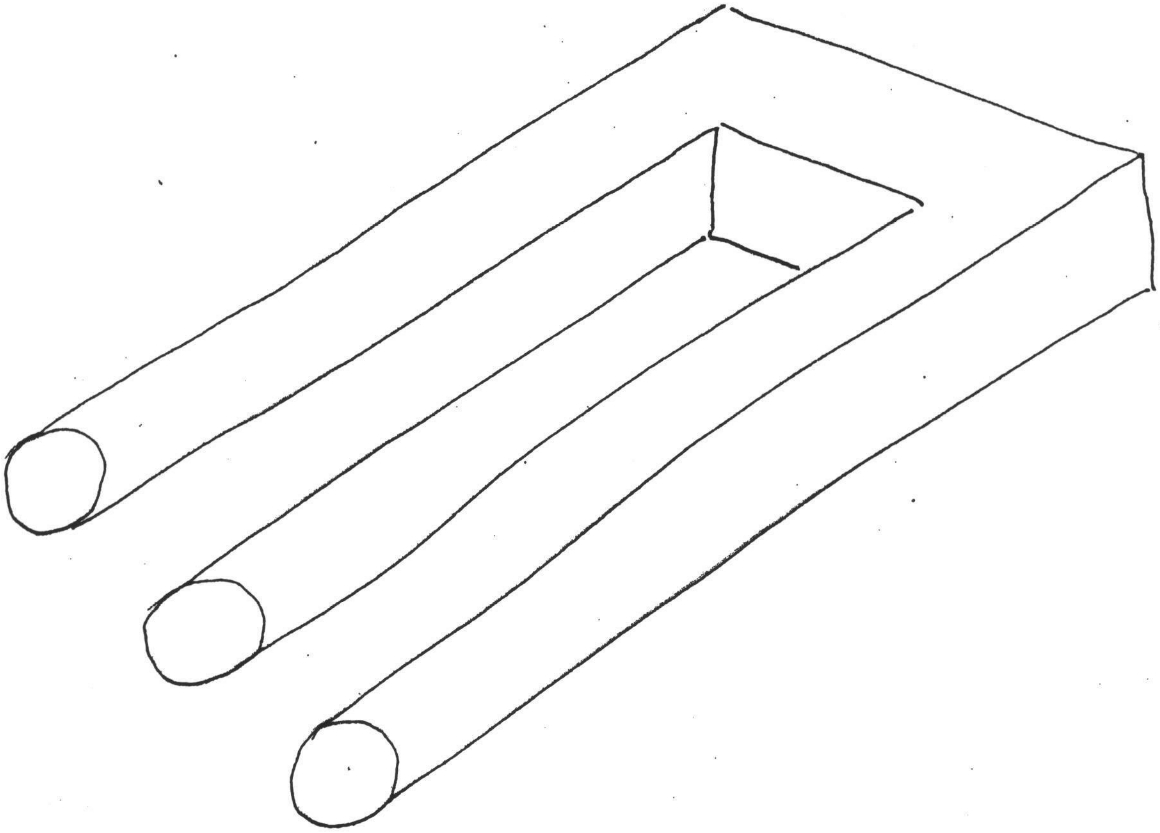
Microarchitecture Structures

- *Evolution of the Process*
- *Branch Prediction*
- *Trace Cache*
- *Block-structured ISA*
- *SSMT – also called "helper" threads (MT → SMT → SSMT)*
- *L2 miss handling*

Evolution of the Process

- *Pipelining*
- *Pipelining (with hiccups)*
- *Wide-issue*
- *Serious Branch Prediction*
- *Speculation …ergo: Recovery*
- *Trace Cache*
- *SMT, SSMT*
- *Soft Errors*
- *L2 misses*
  (e.g., run-ahead, KiloInstruction Processors)

Think Outside the Box



A Few Specifics

- *HPS – expanded on Tomasulo*

- *SMT – expanded on Burton*

- *Perceptron predictor – expanded on Widrow/Rosenblatt/etc.*

The Conditional Branch Problem

* ***Why? Because there are pipelines.***
(Note: HEP)

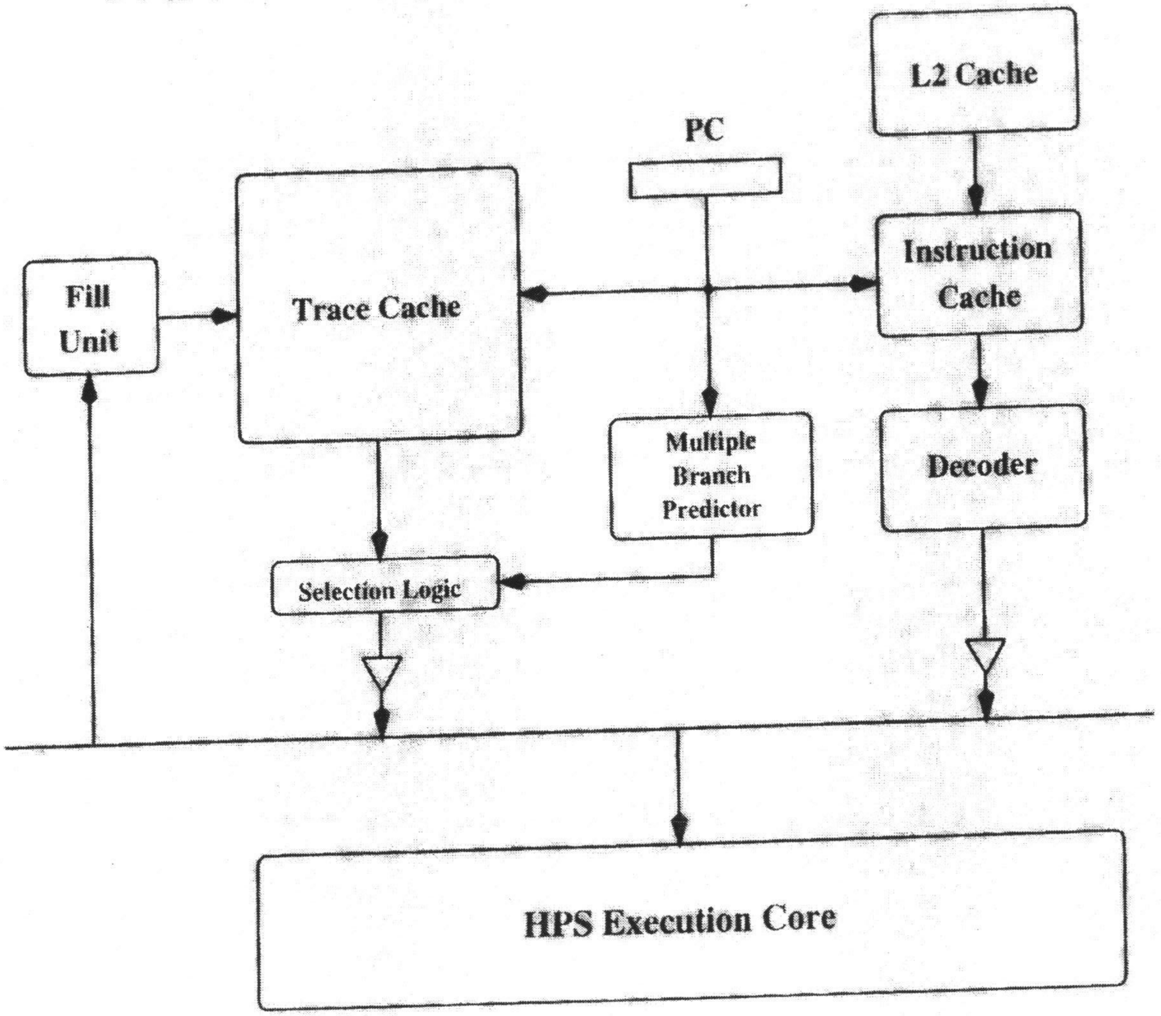
* ***Mechanisms to solve it***

-- Delayed branch
-- Take both paths (SMT variant)
-- Eliminate branches (predication, RS6000)

* ***Branch Prediction***

-- static/dynamic
-- Static: Always taken, BTFN
-- Dynamic: LT, 2bit counter
-- 2 level
-- gshare (first to note interference)
-- agree predictor
-- hybrid
-- indirect jumps
-- return address stack
-- perceptron
-- expose brpred hdware to software
-- wish branch

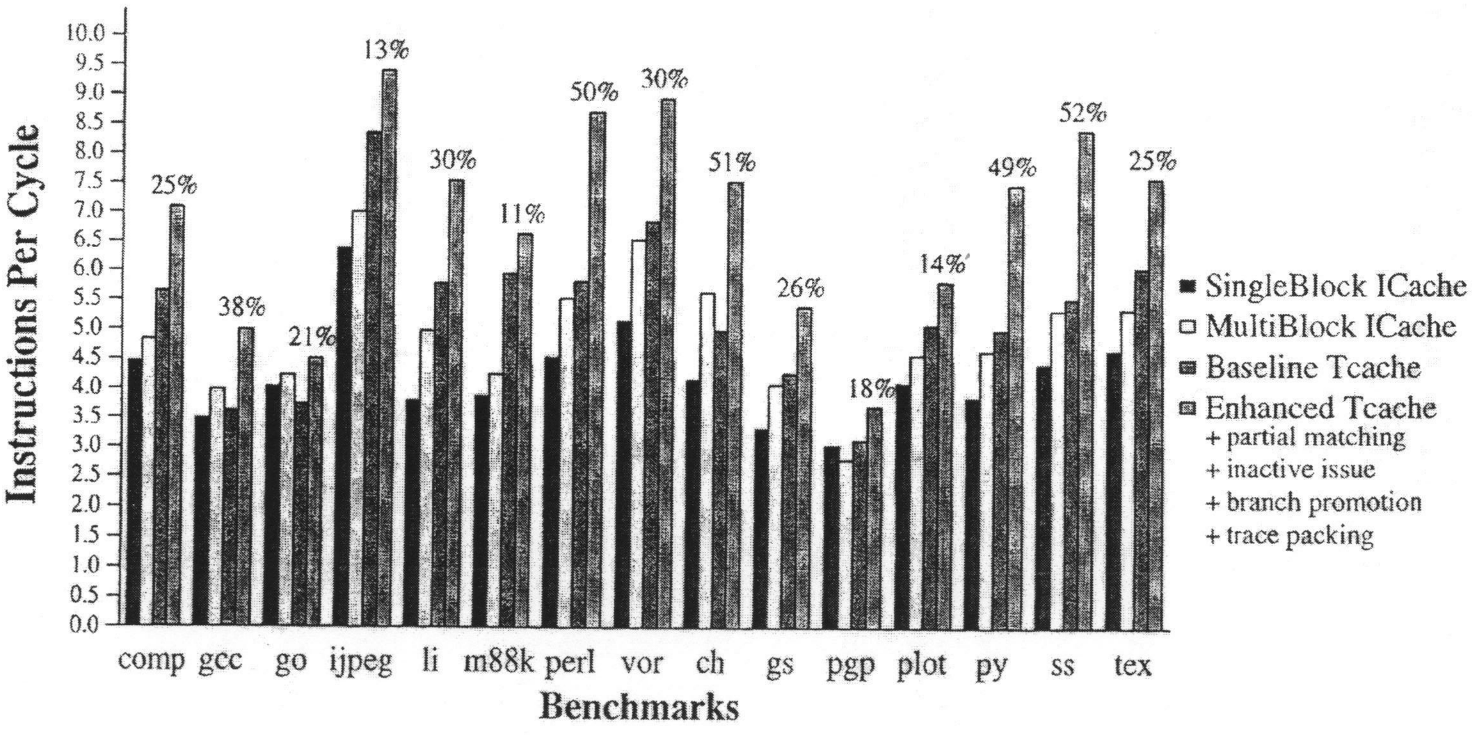
Trace Cache Fetch Mechanism



Trace Cache Issues

- Storage partioning
- Set Associativity
- Path Associativity
- Block Collection (retire vs. issue)
- Fill Unit Latency
- Partial Matching
- Inactive Issue
- Dual Path Segments
- Branch Promotion
- Trace Packing

Overall Performance (aggressive core)

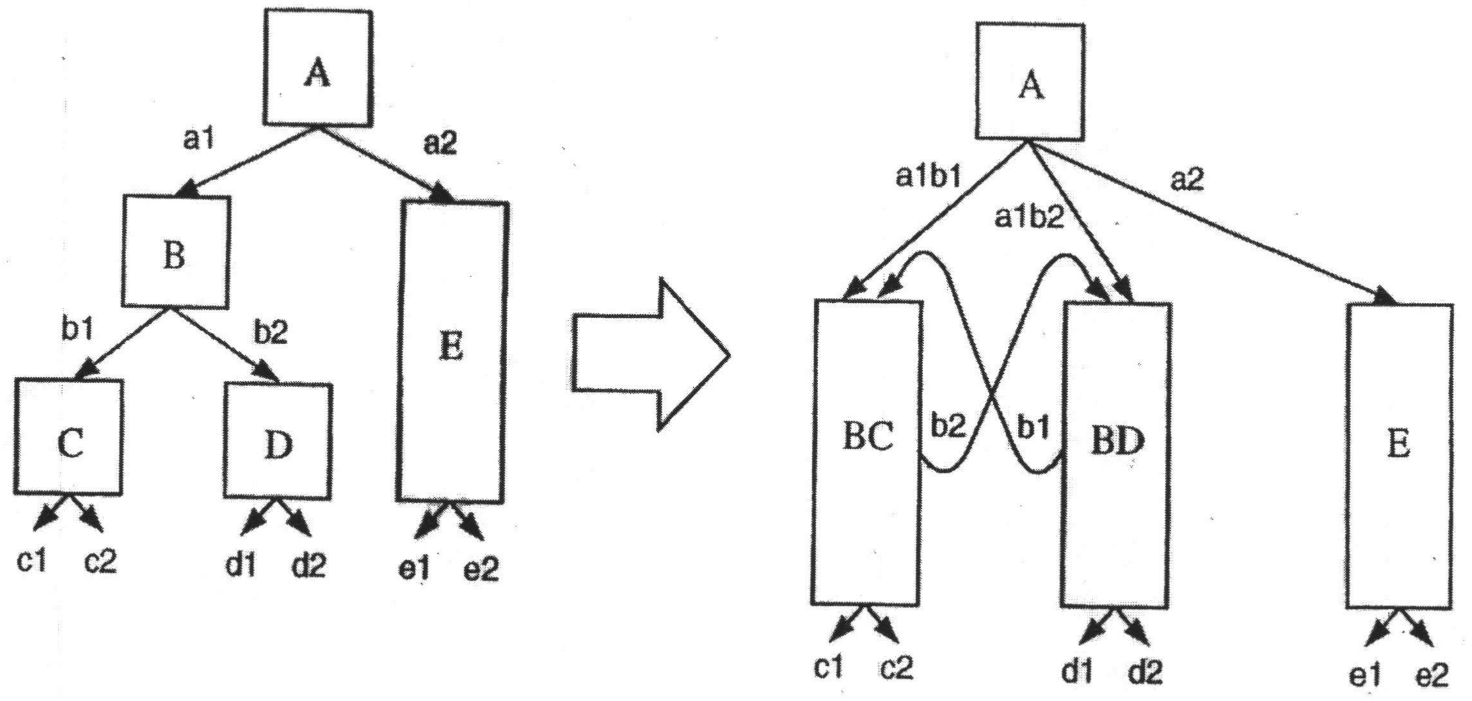


Block-Structured ISA

- ***References:***
  - ***-- Stephen Melvin (PhD, Berkeley, 1991)***
  - ***-- Eric Hao (PhD, Michigan, 1997)***
  - ***-- ISC paper, Melvin and Patt, 1989***
  - ***-- ISCA paper, Melvin and Patt, 1991***
  - ***-- Micro-29, Hao, Chang, Evers, Patt, 1996***

- ***The Atomic Unit of Processing***
- ***Enlarged Blocks***
- ***Savings on Register Pressure***
- ***Faulting Branches, Trapping Branches***
- ***Serial Execution on Exception***
- ***Comparison to Superblocks, Hyperblocks***

Enlarging Basic Blocks



Memory

The Problem: Processor/Memory Imbalance (One of the top two uniprocessor problems)

* MLP replacement policy

* Run-ahead Execution

* Flea-flicker Multipass Execution

* V-way cache

* Other:

-- Address, value prediction
-- Processor in memory
-- SSMT
-- Denser encoding of instructions, data
-- two-level register file
-- better organization of code, data