CS 597d - Advanced Topics in Computer Science: Synergistic Hardware-Compiler Architecture Design

Professor David August

Fall, 1999

Outline

- Administrative Matters
- Course Outline/Subject Overview
- Opportunities
- Who am I?
- Who are you?

Administrative Matters

Lectures:

- Time: MW 11-11:50 or MW 11-12:20
- Room: CS302

Instructor:

- Prof. David August
- Office: CS407
- Phone: 258-2085
- Email: august@cs.princeton.edu
- Office hours: By appointment

Course Web Page:

For announcements, lecture notes, reading list, and project information:

`http://www.cs.princeton.edu/courses/archive/fall99/cs597d/`

More Administrative Matters

Prerequisites:

- Computer Architecture (CS471 or equivalent)
  - assembly language
  - processor pipelining, reason for branch prediction
- Compiler Design (CS320 or equivalent)
  - the C programming language
  - control flow graphs
  - data dependence graphs

If you have any questions or concerns about the prerequisites, please see me.

The Reading

Reading

- Each lecture will have *assigned* and *recommended* papers.
- List of papers is available at the course web page.
- The papers will be available from Tina McCoy in room CS410.

Observations

- Submit observations of one of the *assigned* papers by email to instructor before the associated lecture.
- Submissions will be made available publicly for the benefit of the class.
- An observation is any comment about the paper which indicates that it was read and understood.
  - Comparison to other related work
  - Weakness or strength of paper
  - New ideas provoked
  - Questions raised

The Reading (continued)

Motivation

- Encourage timely exploration of papers
- Help seed ideas for the project
- A mechanism for the exchange of viewpoints

Project and Lectures

You can choose to collaborate in the class project, to give two lectures, or to do both.

Project

- There will be a single group class project.
- The project will be done in collaboration with the instructor.
- Project meeting times will be set by the participants.
- Project will involve learning parts of the IMPACT Research Compiler
- Goal is to generate a publishable work before the end of the semester.

Lecture

- Some lectures may be given by invited guests.
- You can choose to give some as well.
- The topic can be as listed on the syllabus.
- Alternatively, it can be on any topic relevant to compiler/hardware interface issues.

Participation and Grading

Participation

- Effective listening
- Insightful comments
- Meaningful questions

Grading

- Each Lecture: 25%
- Project: 50%
- Readings: 40%
- Participation: 10%

If both the project and a lecture are done, the more favorable grade is used.

The Product

Computers are becoming ubiquitous.

- Mainframes $\rightarrow$ Workstations $\rightarrow$ Desktops $\rightarrow$ Laptop
- Handheld $\rightarrow$ Worn $\rightarrow$ Implanted?

Key metrics of technological progress:

- Power
- Size
- Performance
- Cost
- Reliability

The Artificial Partition

Algorithm | Language | Compiler | ISA | HW Architecture | Circuit | Physical

Compiler:

- Software product division (Microsoft)
- Developed after ISA is defined.
- Computer Scientists

Hardware:

- Hardware products division (Intel)
- Development defines ISA.
- Electrical Engineers

Are there any power, size, performance, cost, or reliability reasons for this partition?

Division of Labor

Instruction Scheduling

- Hardware: few registers in x86 limited static scheduling.
- Compiler: VLIW dynamic scheduling infeasible.

Branch Prediction

- Branch prediction was done in the compiler for lack of better technology.
- Hardware techniques developed to capture dynamic behavior.

Division of labor decisions were not based on performance.

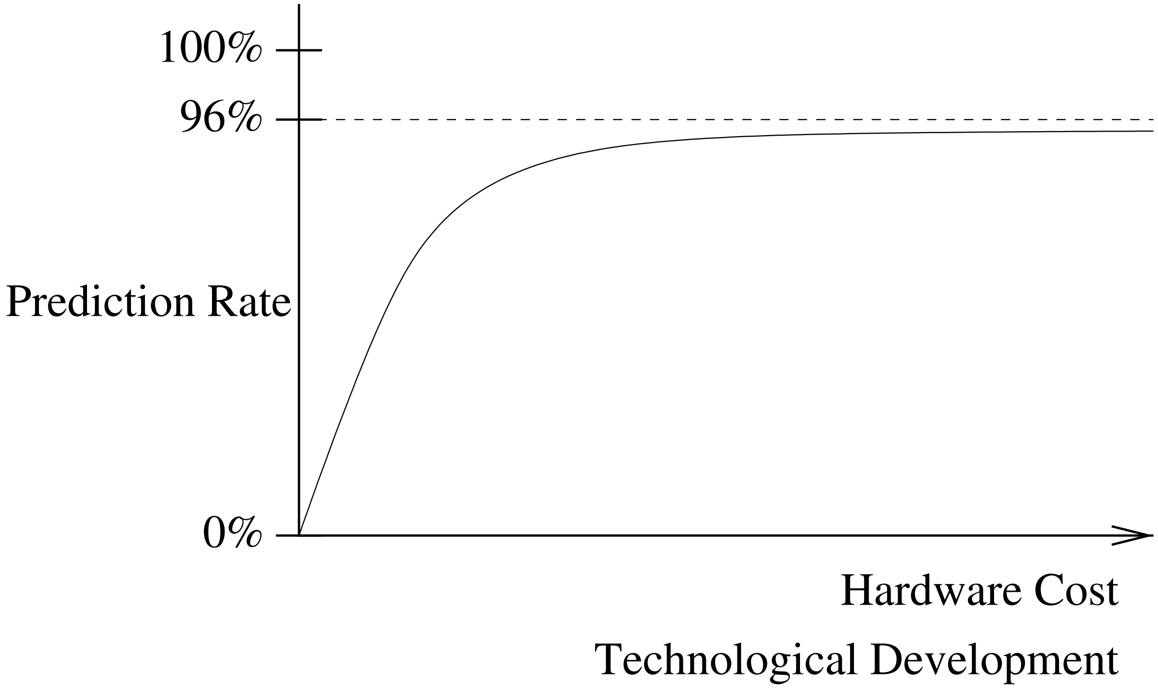
Compiler Limitations

- Compiler analysis is often limited to program inspection.
- Compiler's ability to affect run time is often limited to program transformation.
- Consider Compiler-Based Branch Prediction:
  - Forward Taken, Backward Not Taken - FTBNT
  - Prediction is determined statically at compile time.
  - Prediction doesn't change with respect to differing input sets.
  - Prediction information is fed to the hardware by indirect means.
  - This scheme yields poor performance - not used in modern processors.
  - Compiler cannot express state relationships to the hardware
    - branch correlations
    - other state

What about hardware techniques?

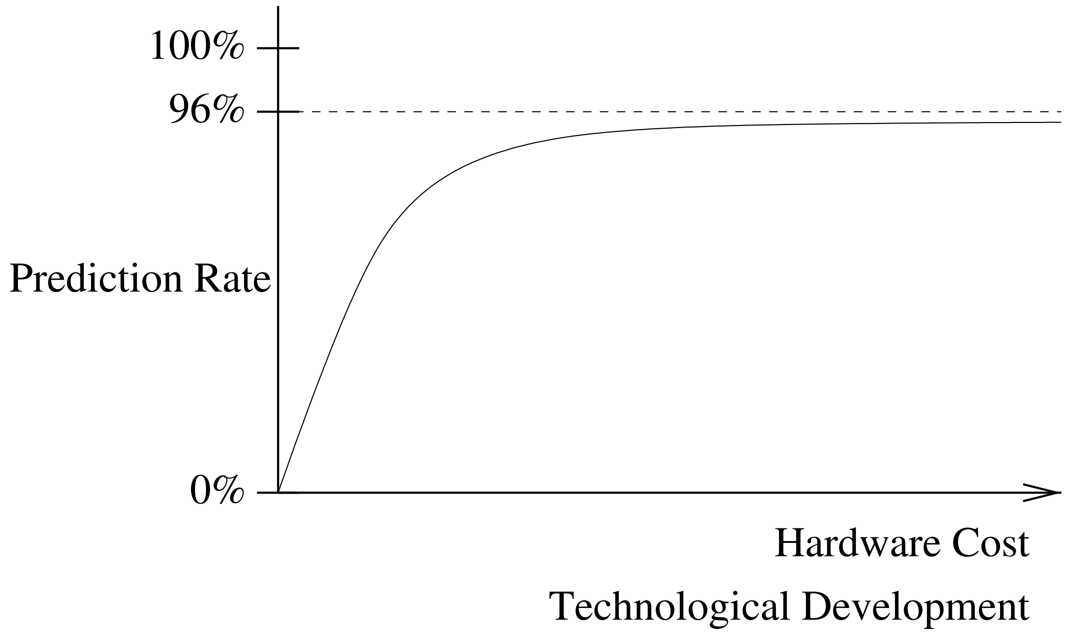
Diminishing Returns

- Hardware-based techniques have had diminishing returns.
- Consider Hardware Branch Prediction:
  - Almost all mechanisms use only branch correlation information.



Misprediction penalties have increased...

Branch Prediction and Information Theory



Information Theory

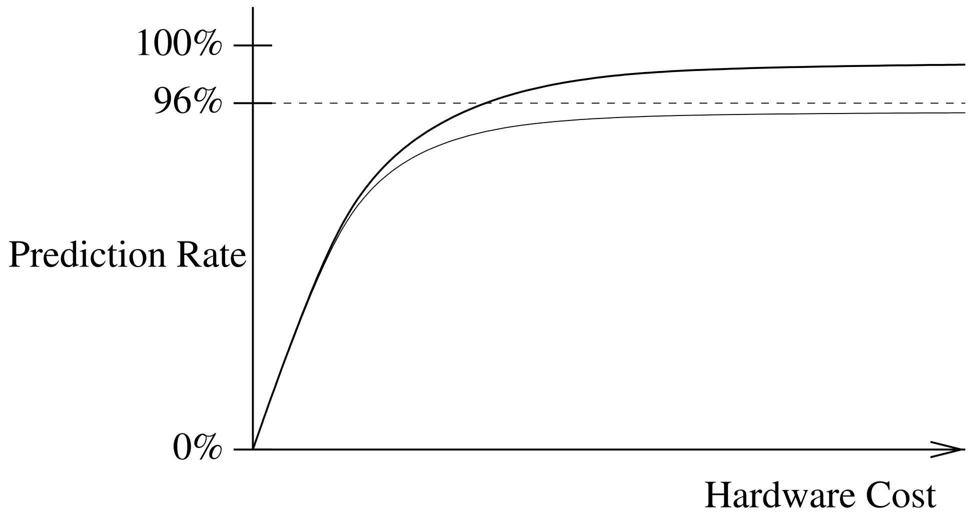
- Predictability is related to entropy.
- You can't predict the next white noise sample.
- Branch history, used by almost all branch predictors, has entropy.
- Branch prediction using branch history has a theoretic limit.
- Trends indicate most prediction information extracted.

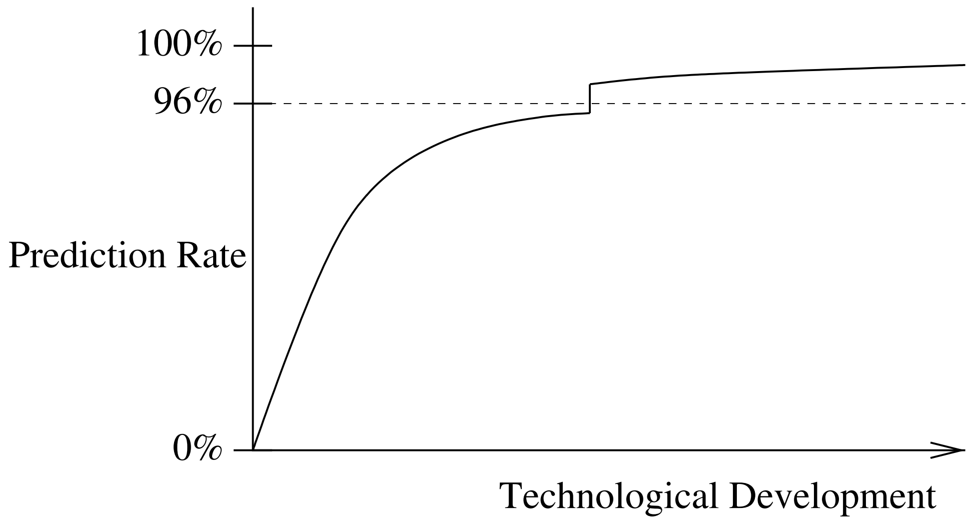
Solution: Alternate Correlation Information Source

	Compiler	Hardware
Static	BTFNT	Taken
Dynamic	Unexplored!	2-Bit Counter 2-Level History

- Compiler Dynamic - compiler points hardware to correlation sources.
- To other architectural state - such as register values.
- This state may be more related to branch outcome.
- Compiler can’t make prediction at compile time.
- Hardware can’t look for correlation in all architectural state.

Solution: Alternate Correlation Information Source





Synergistic Hardware/Compiler interaction breaks prediction barrier...

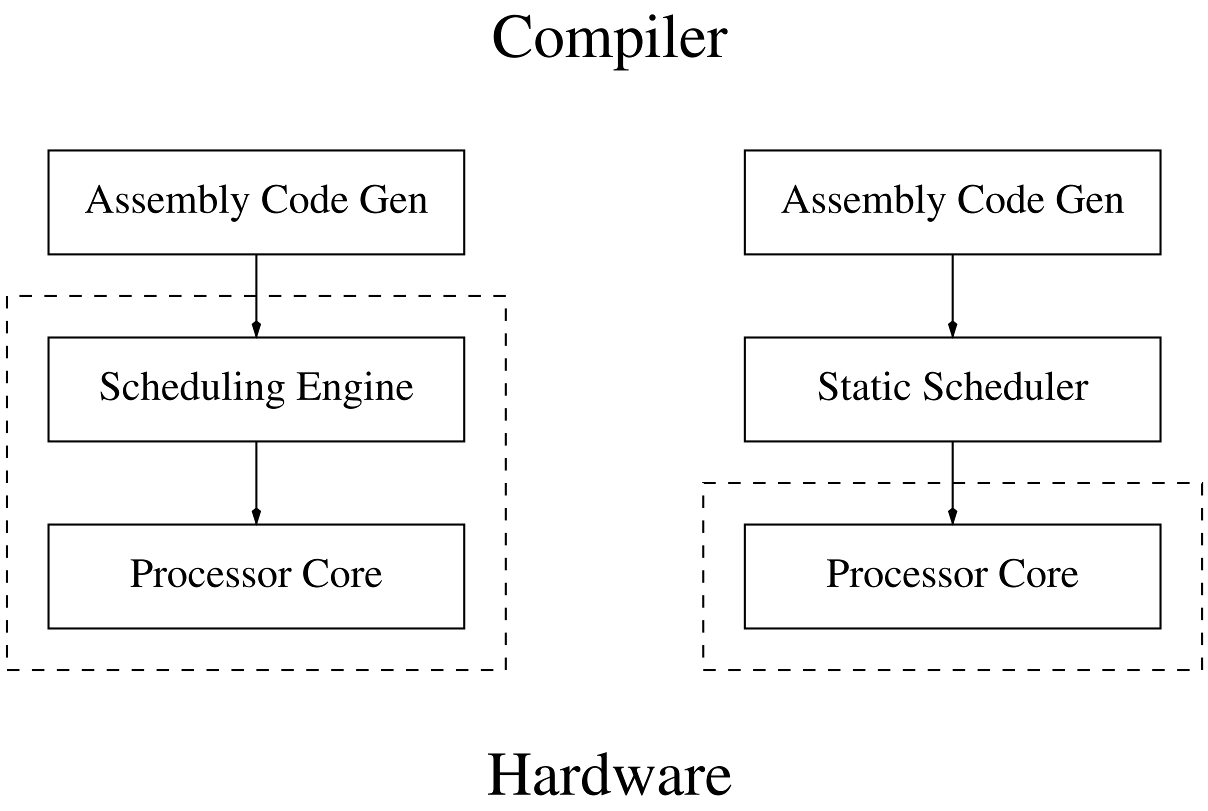
Instruction-Level Parallelism (ILP)

- Instruction-Level Parallelism (ILP), the concurrent execution of independent assembly instructions, is a cost effective way to extract performance from programs.
- Hardware-only scheduling (and speculation) have had diminishing returns.
- Processors such as Pentium III, DEC Alpha 21264 can execute 3 to 6 instructions per cycle, but generally sustain less than 2 instructions per cycle.
- Reasons:
  - Scheduling instruction window is small.
  - Code transformations are not possible.
  - Scheduling at run-time adds complexity and latency to instruction execution.
  - Cache and Branch penalties.

Another approach to ILP → EPIC...

EPIC - Explicitly Parallel Instruction Computing

- Synergistic Hardware/Compiler ILP Instruction Scheduling
- Intel/HP IA-64, Lucent Star*Core

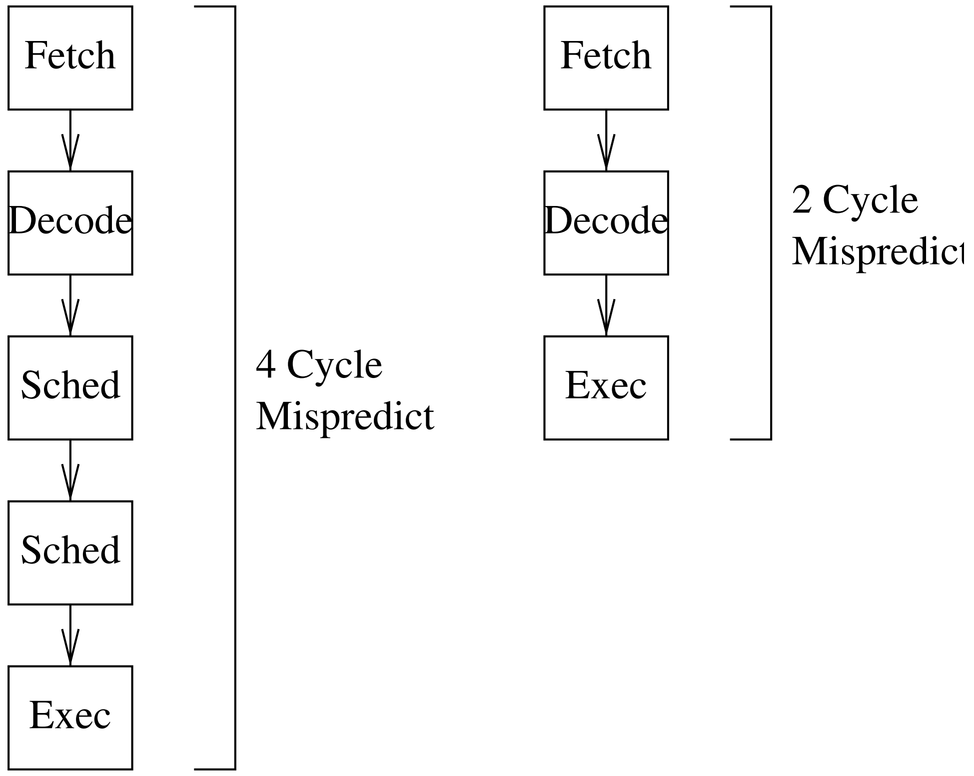


Which uses less power?

EPIC - Explicitly Parallel Instruction Computing

The IA-64 architecture provides mechanisms...to enable the compiler to communicate compile-time information to the processor.

- IA-64 Application Developer's Architecture Guide



Effect on branch prediction? Isn't this VLIW?

Compiler vs. Hardware

In ILP extraction, Branch Prediction, Cache Management, Profiling, and more, the compiler and hardware should each do what they do best.

Processor

- Should only do things which are unique to each execution of the program
- Architecture can see patterns in input data, cache or branch prediction behavior.
- Less is more: Power, Cost, Performance, Size, Reliability
- Reliability: difficult to fix after deployment (FDIV Bug)

Compiler

- Compiler can transform code significantly.
- Compiler can see entire program.
- Do the slow and/or complex analysis - run once.
- Compiler cannot deal with input variation.
- Reliability: can be fixed after deployment (gcc 2.8.2)

Course Outline

- Branch Prediction - The compiler helps hardware break the barrier.
- Profiling - The hardware gives the compiler eyes.
- EPIC - A new approach to instruction level parallelism
  - Control Speculation
  - Predication
  - Memory Speculation
  - Value Speculation
  - Memory Dependence Analysis
  - Memory Management

Opportunities

- Companies are looking for people with an understanding of Compiler/Architecture co-design - Summer Internship and Full-time.
  - Intel, Hewlett-Packard, Lucent
  - Sun, SGI, Microsoft
- The IMPACT Research Compiler
  - Can significantly improve compiler and hardware research.
  - Used and licensed by Intel, Hewlett-Packard, Lucent, Microsoft, Sun, and others.
- There are many questions in this area which remain unexplored.

Who Am I?

Experience

- Intel Corporation, Hillsboro, OR.
  - Multi-processor validation
- Hewlett-Packard Laboratories, Palo Alto, CA.
  - IMPACT and Elcor research compilers
- Intel Corporation, Santa Clara, CA.
  - IMPACT for IA-64
- Lucent Technologies, Allentown, PA.
  - Compiler for EPIC processor

Who Are You?

- Full Name
- Email address
- Department
- Year - Senior, Second year graduate student, etc.
- Level of commitment/interest in course - Taking for grade, Audit, etc.
- Areas of interest
- Textbooks used in Compiler and Architecture classes
- Relevant previous experience
- Anything else?

Lecture 2 Reading

For Lecture 2 (9/22/99) - Read All, Observe 1:

"Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation", Shien-Tai Pan, Kimming So, Joseph T. Rahmeh, *Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems.*

"Alternative Implementations of Two-Level Adaptive Branch Prediction", Tse-Yu Yeh and Yale N. Patt, *Proceedings of the 19th International Symposium on Computer Architecture*.

"Alternative Implementations of Hybrid Branch Predictors", Po-Yung Chang, Eric Hao, Yale N. Patt, *Proceedings of the 28th International Symposium on Computer Architecture*.

Lecture 2 Reading Continued

For Further Exploration:

"Branch Prediction Based on Universal Data Compression Algorithms", Eitan Fedorovsky, Meir Feder, Shlomo Weiss, *Proceedings of the 25th International Symposium on Computer Architecture*.

"A Language for Describing Predictors and its Application to Automatic Synthesis", Joel Emer and Nikolas Gloy, *Proceedings of the 24th International Symposium on Computer Architecture*.