Branch Prediction and Instruction Prefetch

Prof. Scott Rixner
Duncan Hall 3028
rixner@rice.edu

January 18, 2007

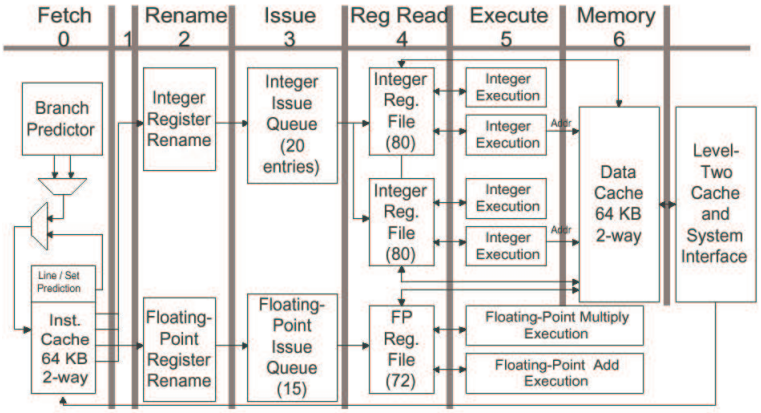
Pipeline Stalls

- **Data dependencies**
- **Resource conflicts**
- **Fetch access delays**
- **Branches**
  - **Discard fetched instructions**
  - **Restart fetching from target**

Branch Prediction

- **What is branch prediction?**
- **What are we predicting?**
- **How does it reduce pipeline stalls?**

Alpha 21264 Pipeline

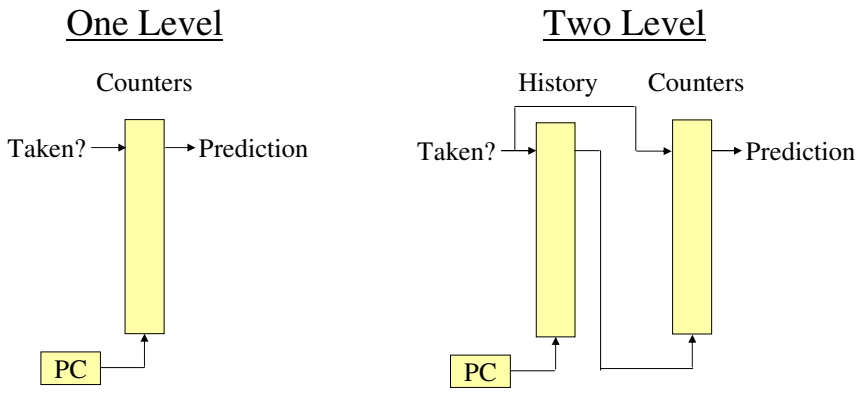


-IEEE Micro 3/99

Prediction Mechanisms

- **Static**
- **Dynamic**
- **Prediction based on opcode**
- **Return address stack (RAS)**

Branch Predictors

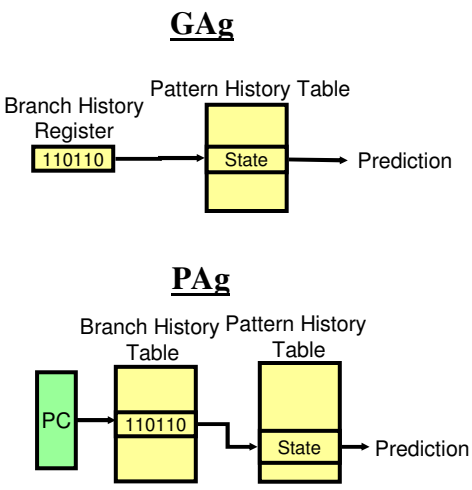


Two-level Branch Prediction

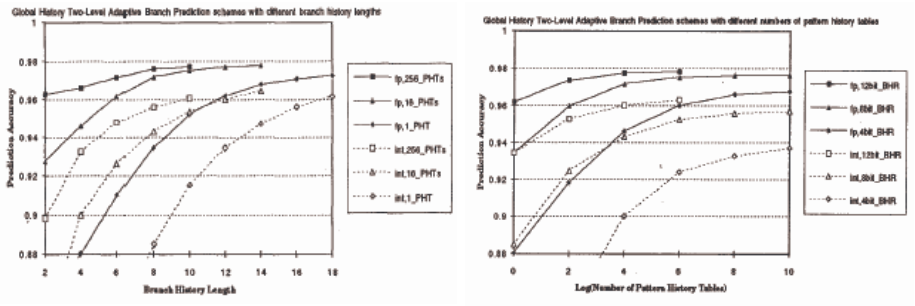
- **Classification of two level branch predictors**
  - **Branch history (G, S, or P)**
  - **Pattern history (g, s, or p)**
  - **9 total predictors**
- **Trace-driven simulation**
  - **Determine branch prediction accuracy**
  - **SPEC89 benchmarks**
- **Cost analysis**
  - **Number of bits required for each predictor**
- **Performance?**

Two Level Predictors

- **GAg**
  - **Global branch history, global pattern histroy**
- **GAs**
  - **Global branch history, per-set pattern history**
- **GAp**
  - **Global branch history, per-address pattern history**
- **PAg**
  - **Per-address branch history, global pattern history**
- **PAp**
  - **Per-address branch history, per-address pattern history**
- **…**

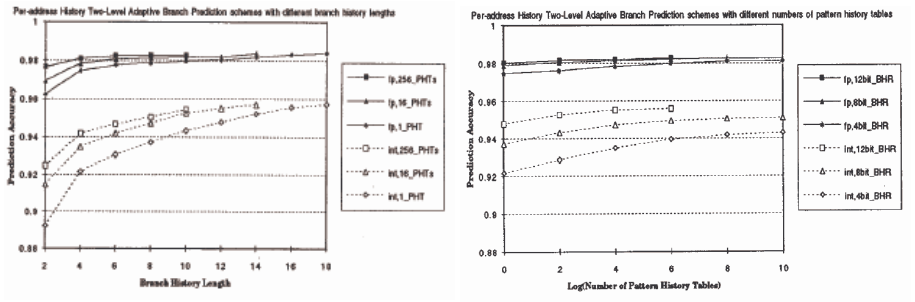


Global History Schemes



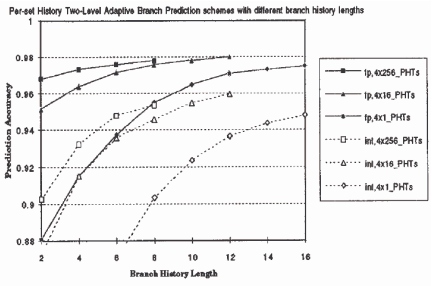
-Yeh and Patt, ISCA '93

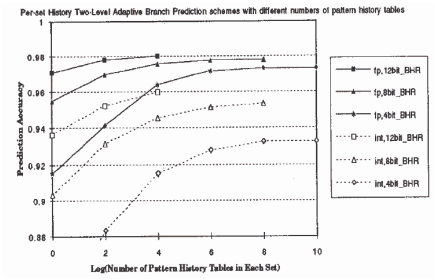
Per-address History Schemes



-Yeh and Patt, ISCA '93

Per-set History Schemes





-Yeh and Patt, ISCA '93

Costs

Scheme Name	History Register Length	Number of Pattern History Tables	Simplified Hardware Cost
GAg(k)	k	1	$k + 2^k \times 2$
GAs(k,p)	k	p	$k + p \times 2^k \times 2$
GAp(k)	k	b	$k + b \times 2^k \times 2$
PAg(k)	k	1	$b \times k + 2^k \times 2$
PAs(k,p)	k	p	$b \times k + p \times 2^k \times 2$
PAp(k)	k	b	$b \times k + b \times 2^k \times 2$
SAg(k)	k	s×1	$s \times k + s \times 2^k \times 2$
SAs(k,s×p)	k	s×p	$s \times k + s \times p \times 2^k \times 2$
SAp(k)	k	s×b	$s \times k + s \times b \times 2^k \times 2$

b is the number of entries in the BHT and s is the number of branch sets.

Table 3: Conditional branch predictor configurations and their estimated costs.

-Yeh and Patt, ISCA '93

Per-address "Cost Effectiveness"

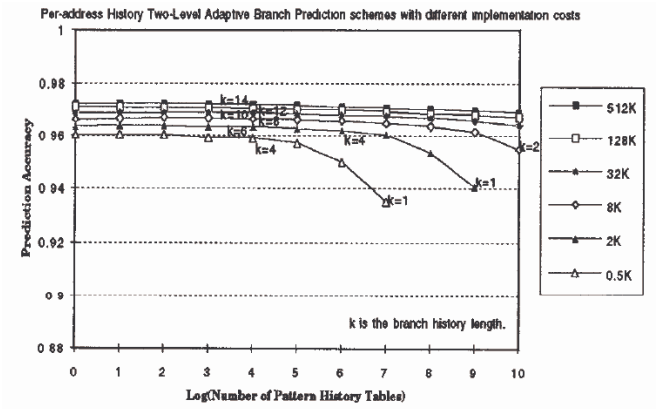


Figure 10: Per-address history schemes with different implementation costs.

-Yeh and Patt, ISCA '93

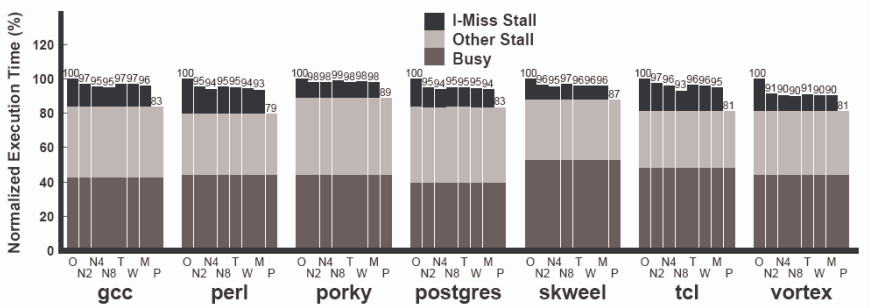
Prefetching

- **Why prefetch?**
- **Terminology**
  - **Coverage factor**
  - **Unnecessary prefetch**
  - **Useless prefetch**
  - **Prefetching distance**

Instruction Prefetching

- **Next-N-line**
  - **Each fetch also prefetches N sequential cache lines**
- **Target-line**
  - **Predict next line and prefetch it**
- **Wrong-path**
  - **Next-N-line combined with prefetching of lines targeted by static branches**
- **Markov**
  - **Use miss address prediction table to prefetch lines targeted by dynamic branches**

Hardware Prefetching Effectiveness

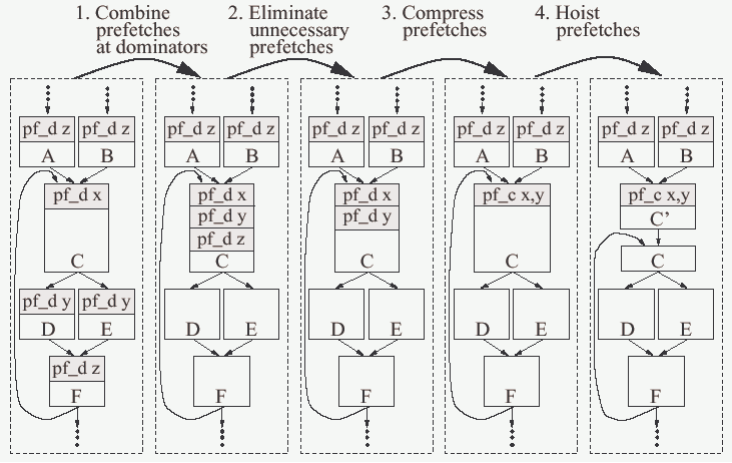


- **Why are these schemes not more effective?**
- **Is that an inherent problem with hardware prefetching?**

Hardware Mechanisms

- **Instruction-prefetch instructions**
  - **Compiler provided hints allow prefetching**
  - **Dropped after decode stage (does not 'execute')**
- **Prefetch filtering**
  - **Between I-prefetcher and L2 cache**
  - **Reduces the number of useless prefetches**
  - **L2 line counters increment for unused prefetches**
  - **Prefetches are ignored for large counts**

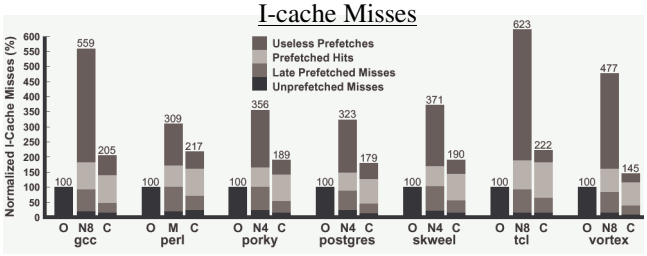
Compiler Support



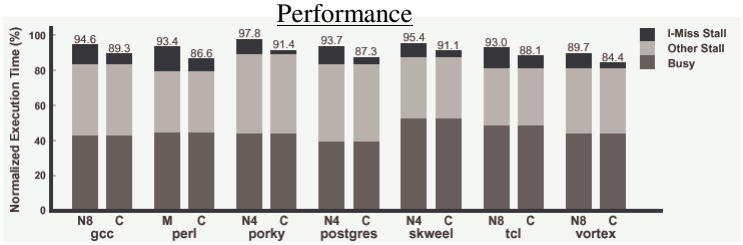
-Luk & Mowry 12/98

Results

I-cache Misses



Performance



-Luk & Mowry 12/98

Next Class

- **Tuesday – Scalable Instruction Fetch and Trace Caching**
  - **"A Scalable Front-End Architecture for Fast Instruction Delivery"**
  - **"Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching"**