Advanced Branch Prediction

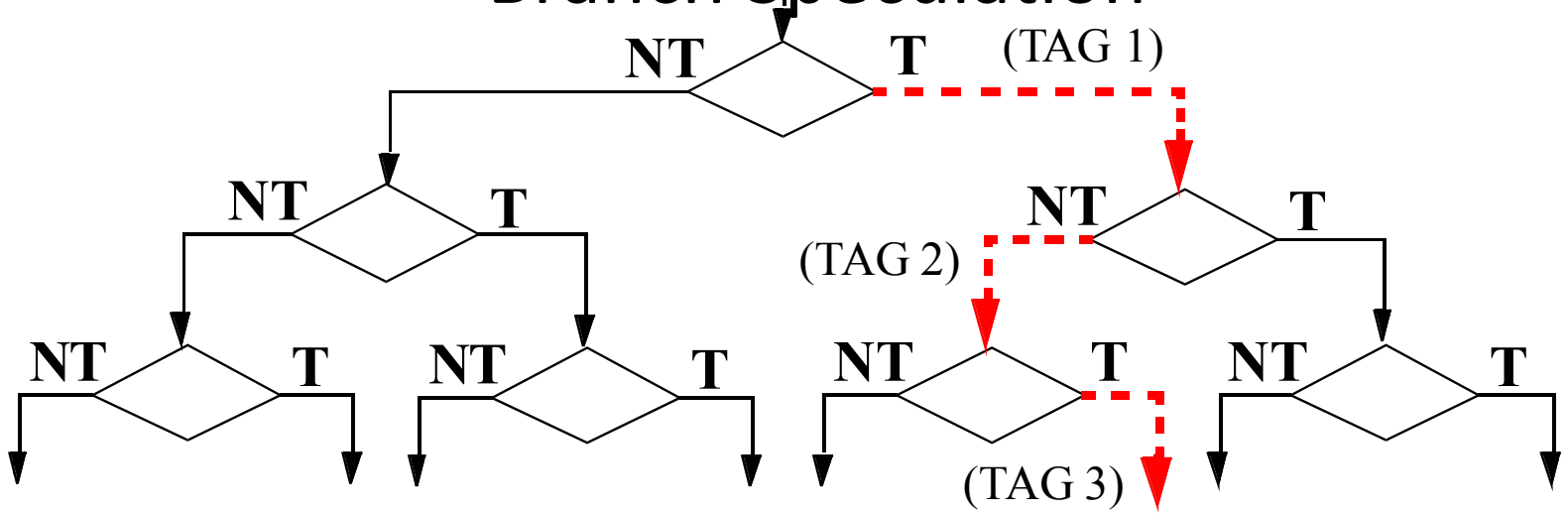
Prof. Mikko H. Lipasti
University of Wisconsin-Madison

Lecture notes based on notes by John P. Shen
Updated by Mikko Lipasti

Advanced Branch Prediction

- Control Flow Speculation
  - Branch Speculation
  - Mis-speculation Recovery
- Branch Direction Prediction
  - Static Prediction
  - Dynamic Prediction
  - Hybrid Prediction
- Branch Target Prediction
- High-bandwidth Fetch
- High-Frequency Fetch

Branch Speculation

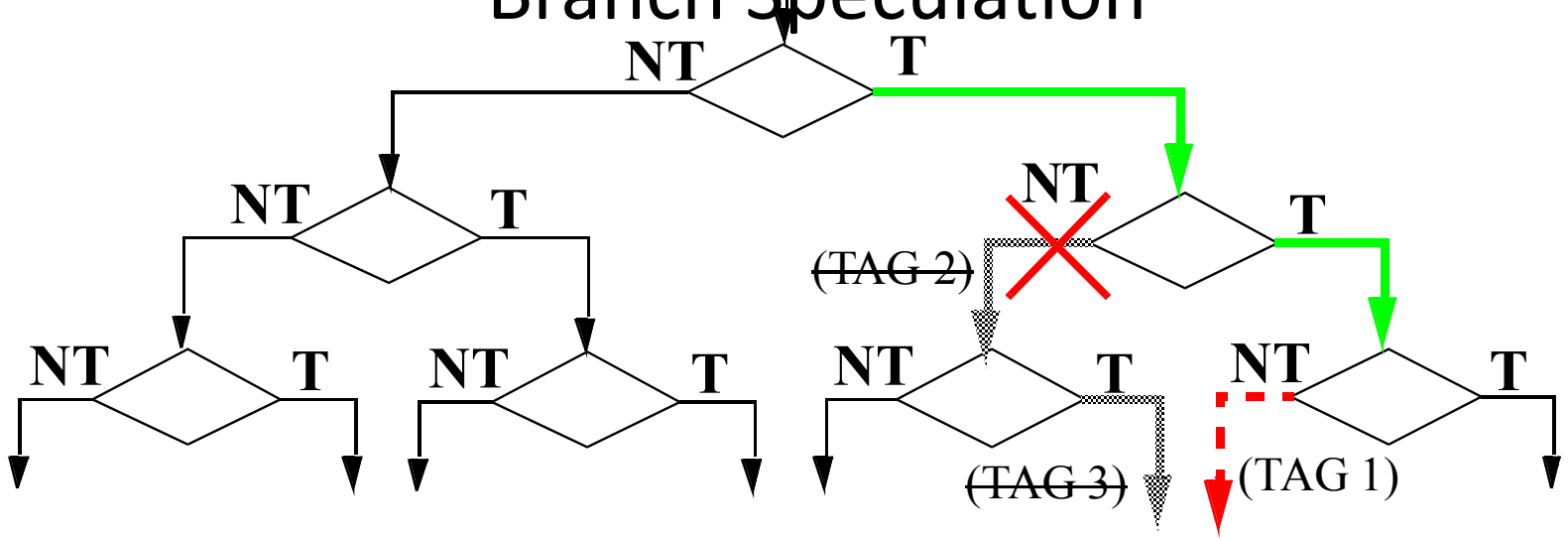


- Leading Speculation
  - Typically done during the Fetch stage
  - Based on potential branch instruction(s) in the current fetch group
- Trailing Confirmation
  - Typically done during the Branch Execute stage
  - Based on the next Branch instruction to finish execution

Branch Speculation

- Leading Speculation
  1. Tag speculative instructions
  2. Advance branch and following instructions
  3. Buffer addresses of speculated branch instructions
- Trailing Confirmation
  1. When branch resolves, remove/deallocate speculation tag
  2. Permit completion of branch and following instructions

Branch Speculation



- Start new correct path
  - Must remember the alternate (non-predicted) path
- Eliminate incorrect path
  - Must ensure that the mis-speculated instructions produce no side effects

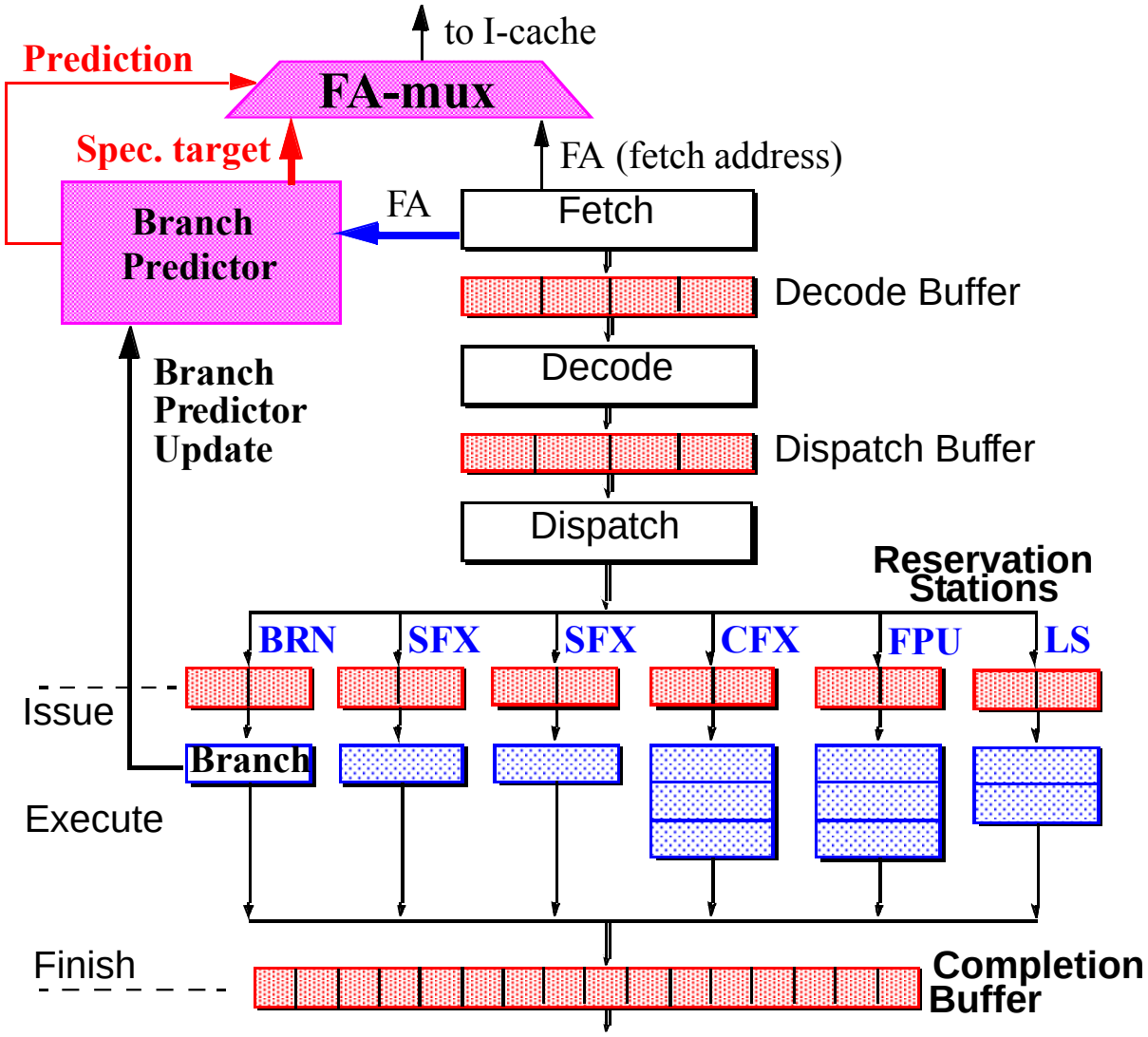
Mis-speculation Recovery

- Start new correct path
  1. Update PC with computed branch target (if predicted NT)
  2. Update PC with sequential instruction address (if predicted T)
  3. Can begin speculation again at next branch
- Eliminate incorrect path
  1. Use tag(s) to deallocate ROB entries occupied by speculative instructions
  2. Invalidate all instructions in the decode and dispatch buffers, as well as those in reservation stations

Tracking Instructions

- Assign branch tags
  - Allocated in circular order
  - Instruction carries this tag throughout processor
- Track instruction groups
  - Instructions managed in groups, max. one branch per group
  - ROB structured as groups
    - Leads to some inefficiency
    - Simpler tracking of speculative instructions

Program Control Flow



Static Branch Prediction

- Single-direction
  - Always not-taken: Intel i486
- Backwards Taken/Forward Not Taken
  - Loop-closing branches
  - Used as backup in Pentium Pro, II, III, 4
- Heuristic-based:

```
void * p = malloc (numBytes);
if (p == NULL)
        errorHandlingFunction( );
```

Static Branch Prediction

Heuristic Name	Description
Loop Branch	If the branch target is back to the head of a loop, predict taken.
Pointer	If a branch compares a pointer with NULL, or if two pointers are compared, predict in the direction that corresponds to the pointer being not NULL, or the two pointers not being equal.
Opcode	If a branch is testing that an integer is less than zero, less than or equal to zero, or equal to a constant, predict in the direction that corresponds to the test evaluating to false.
Guard	If the operand of the branch instruction is a register that gets used before being redefined in the successor block, predict that the branch goes to the successor block.
Loop Exit	If a branch occurs inside a loop, and neither of the targets is the loop head, then predict that the branch does not go to the successor that is the loop exit.
Loop Header	Predict that the successor block of a branch that is a loop header or a loop pre-header is taken.
Call	If a successor block contains a subroutine call, predict that the branch goes to that successor block.
Store	If a successor block contains a store instruction, predict that the branch does not go to that successor block.
Return	If a successor block contains a return from subroutine instruction, predict that the branch does not go to that successor block.

- Heuristic-based: Ball/Larus
  - Thomas Ball and James R. Larus. Branch Prediction for Free. ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pages 300-313, May 1993.

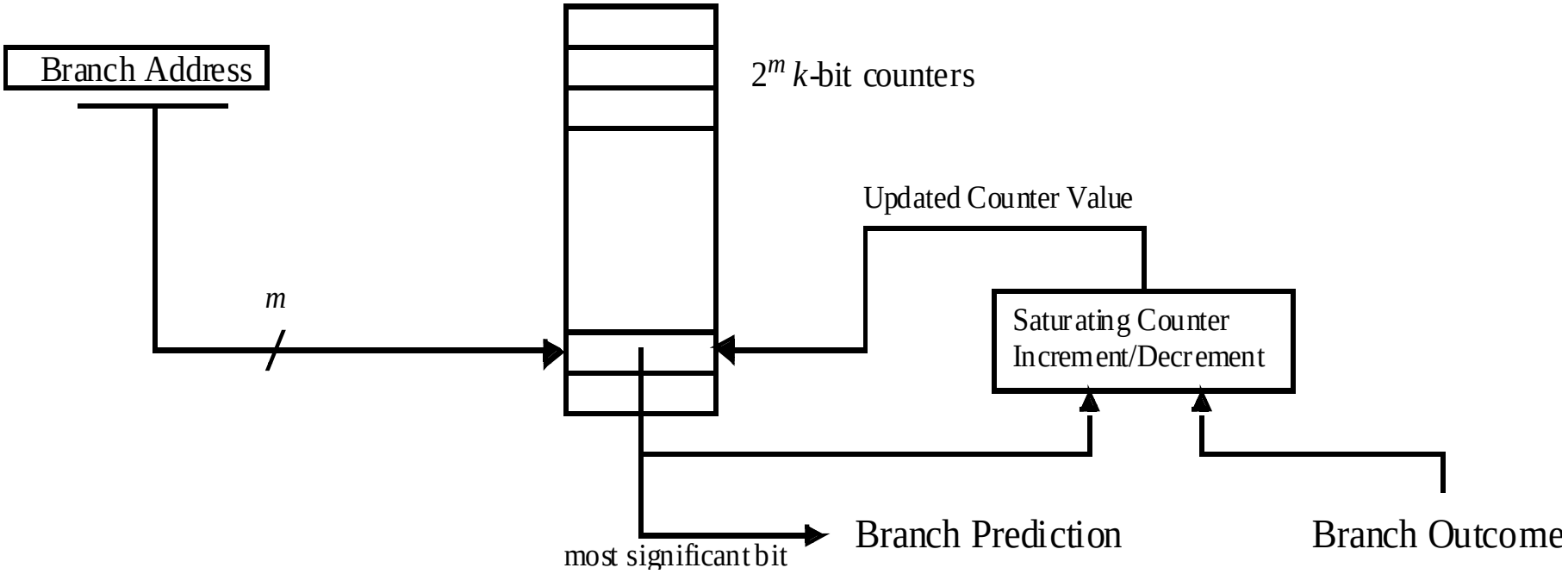
Static Branch Prediction

- Profile-based
  1. Instrument program binary
  2. Run with representative (?) input set
  3. Recompile program
     a. Annotate branches with hint bits, or
     b. Restructure code to match predict not-taken
- Best performance: 75-80% accuracy

Dynamic Branch Prediction

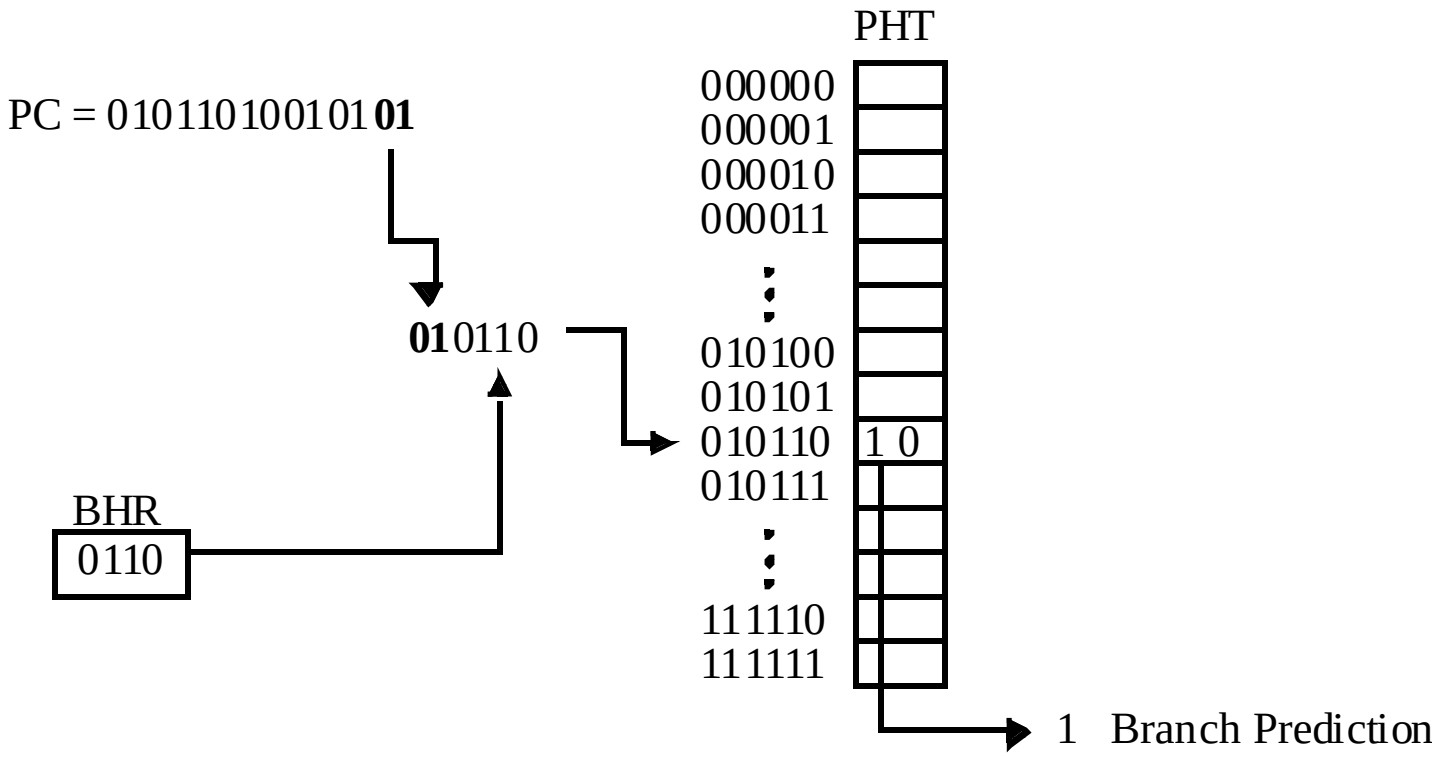
- Main advantages:
  - Learn branch behavior autonomously
    - No compiler analysis, heuristics, or profiling
  - Adapt to changing branch behavior
    - Program phase changes branch behavior
- First proposed in 1979-1980
  - US Patent #4,370,711, Branch predictor using random access memory, James. E. Smith
- Continually refined since then

Smith Predictor Hardware



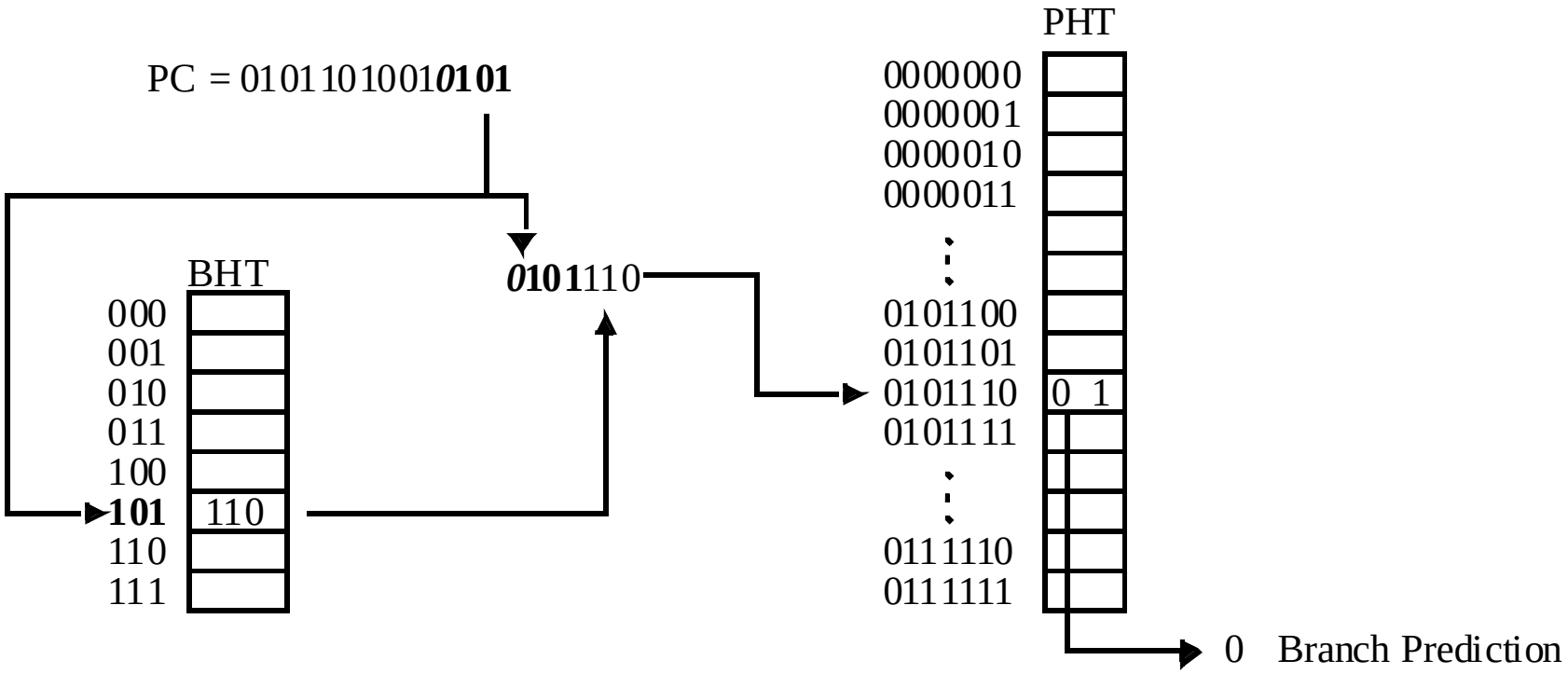
- Jim E. Smith. A Study of Branch Prediction Strategies. International Symposium on Computer Architecture, pages 135-148, May 1981
- Widely employed: Intel Pentium, PowerPC 604, PowerPC 620, etc.

Two-level Branch Prediction



- BHR adds *global* branch history
  - Provides more context
  - Can differentiate multiple instances of the same static branch
  - Can correlate behavior across multiple static branches

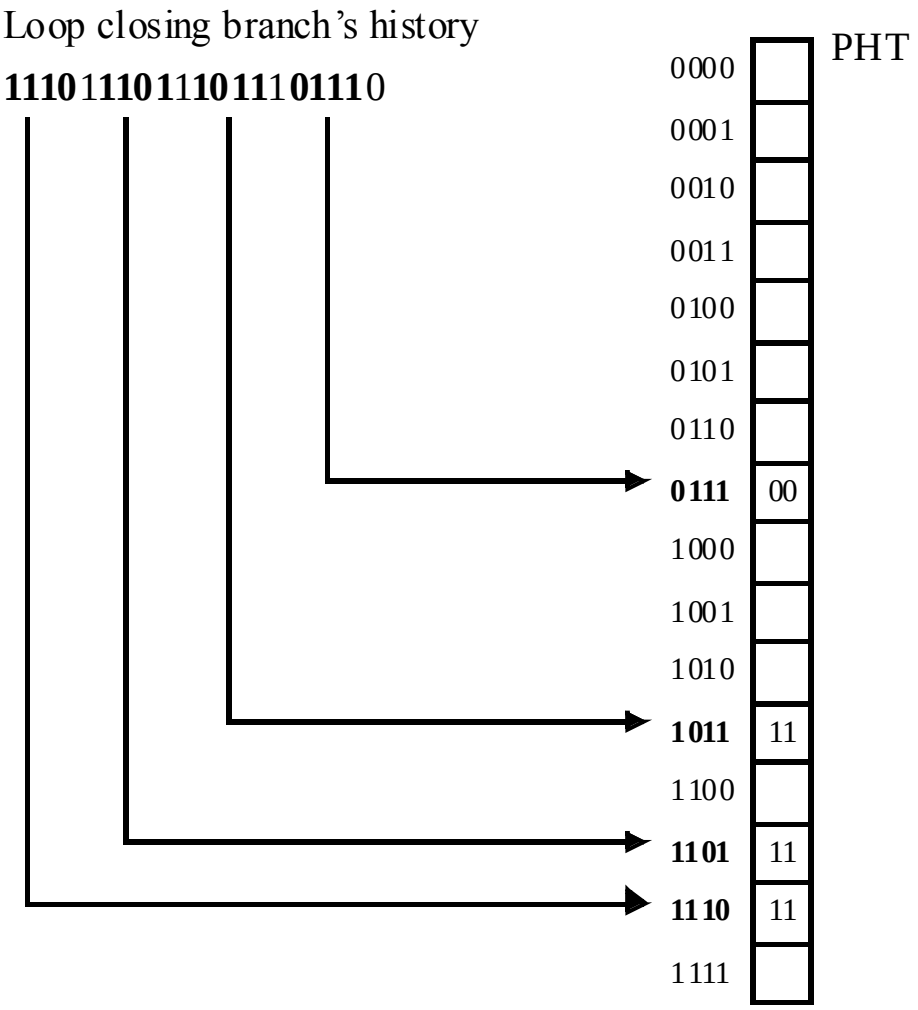
Two-level Prediction: Local History



- Detailed local history can be useful

Local History Predictor Example

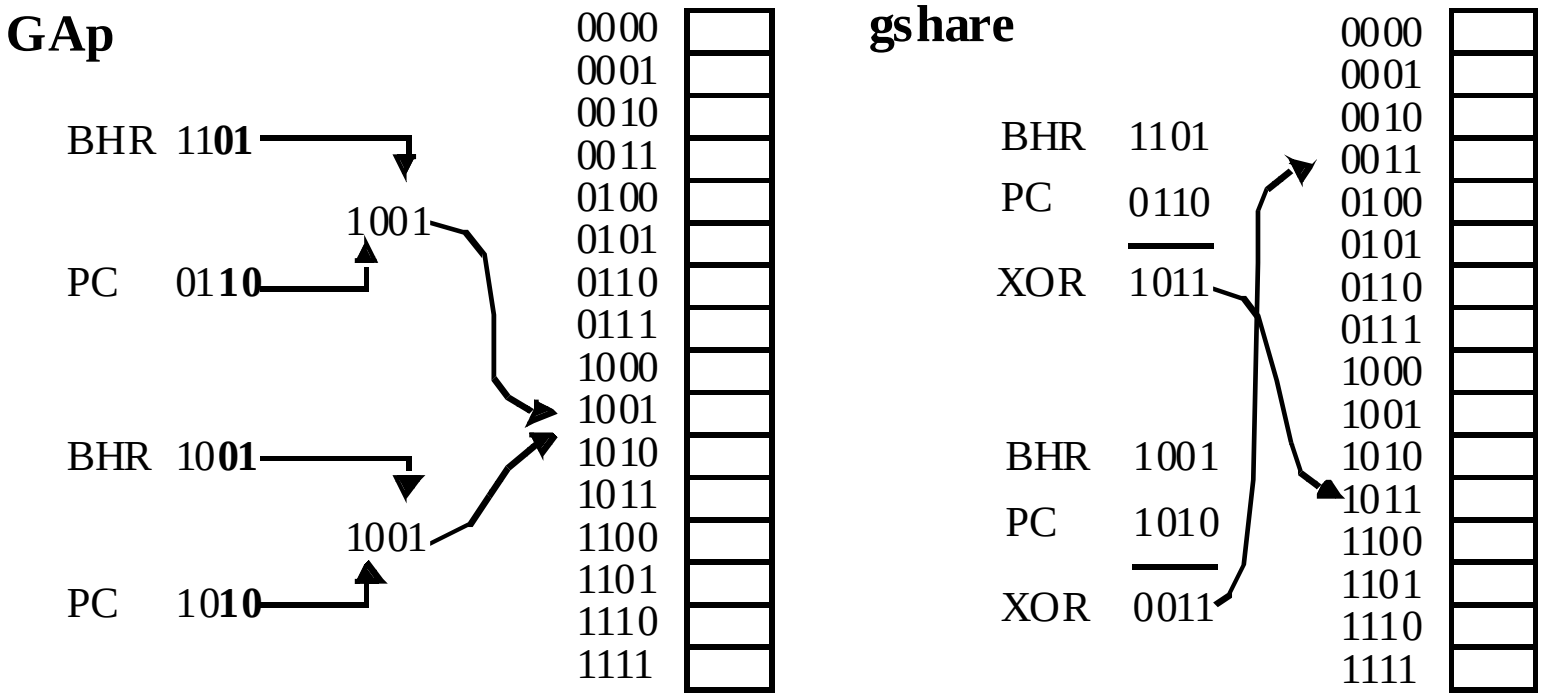
- Loop closing branches
  - Must identify last instance
- Local history dedicates PHT entry to each instance
  - '0111' entry predicts not taken



Two-level Taxonomy

- Based on indices for branch history and pattern history
  - BHR: {G,P,S}: {Global, Per-address, Set}
  - PHT: {g,p,s}: {Global, Per-address, Set}
  - 9 combinations: GAg, GAp, GAs, PAg, PAp, PAs, SAg, SAp and SAs
- Tse-Yu Yeh and Yale N. Patt. Two-Level Adaptive Branch Prediction. International Symposium on Microarchitecture, pages 51-61, November 1991.

Index Sharing in Two-level Predictors



- Use XOR function to achieve better utilization of PHT
- Scott McFarling. Combining Branch Predictors. TN-36, Digital Equipment Corporation Western Research Laboratory, June 1993.
- Used in e.g. IBM Power 4, Alpha 21264

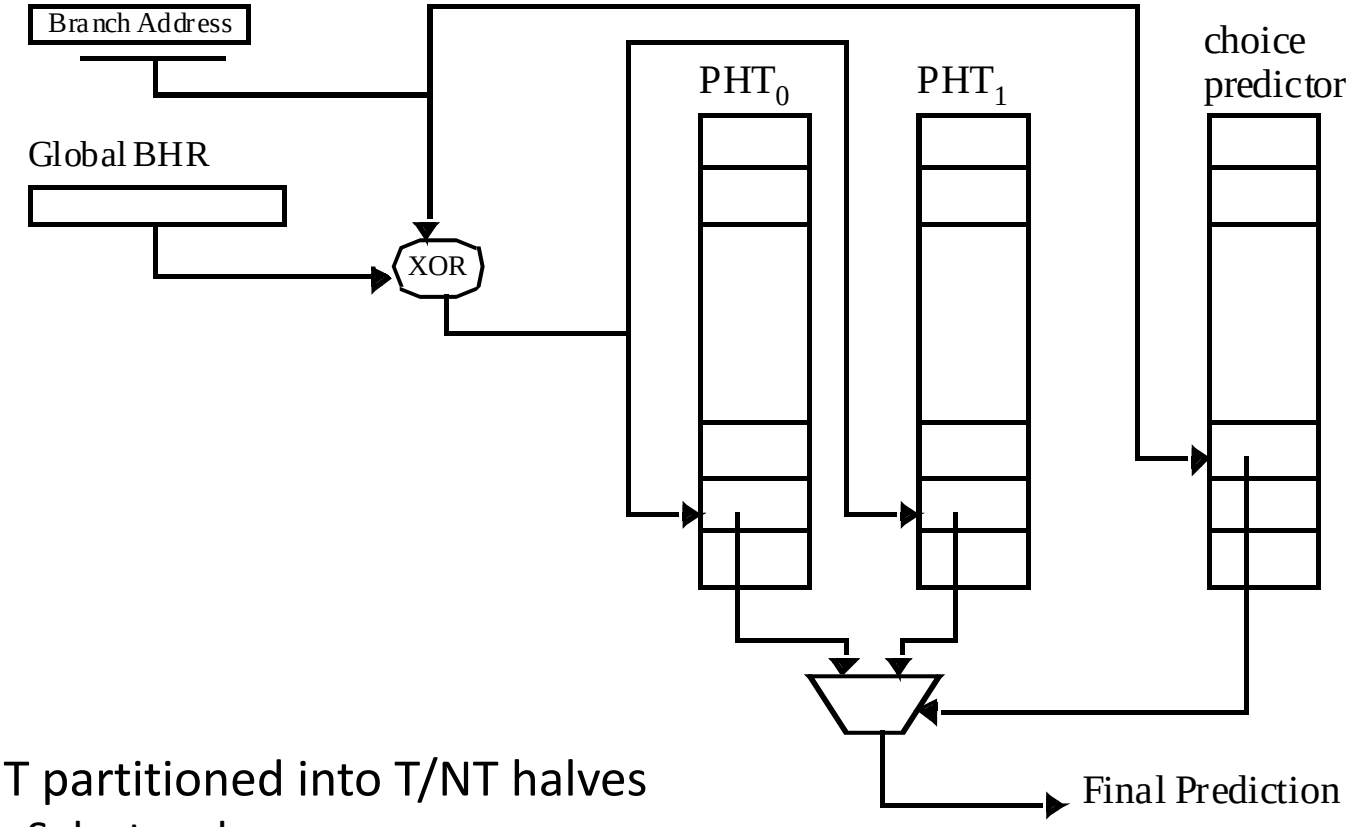
Sources of Mispredictions

- Lack of history (training time)
- Randomized behavior
  - Usually due to randomized input data (benchmarks)
  - Surprisingly few branches depend on input data values
- BHR capacity
  - Correlate to branch that already shifted out
  - E.g. loop count > BHR width
- PHT capacity
  - Aliasing/interference
    - Positive
    - Negative

Reducing Interference

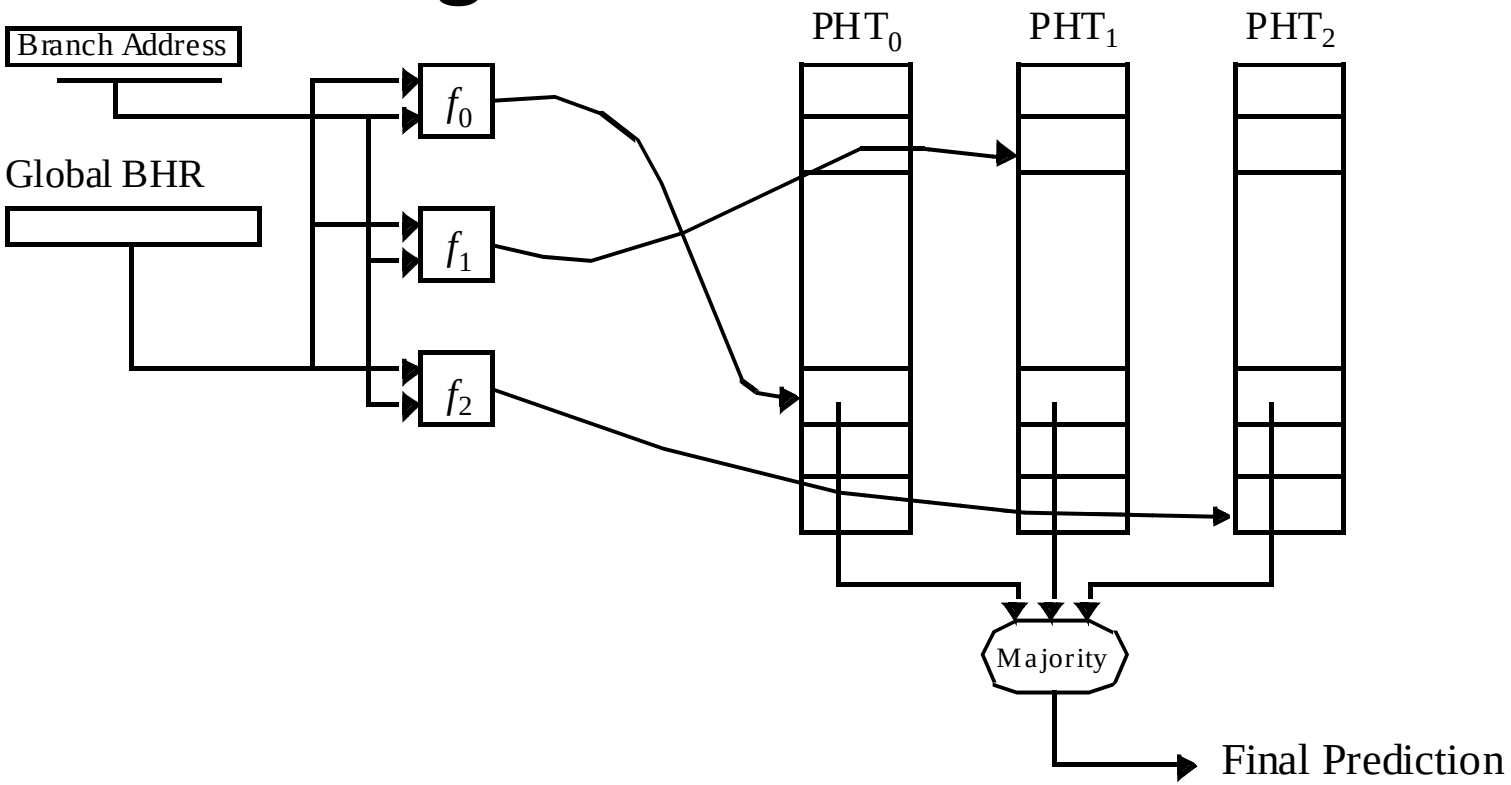
- Compulsory aliasing (*cold miss*)
  - Not important (less than 1%)
  - Only remedy is to set appropriate initial value
  - Also: beware indexing schemes with high training cost (e.g. very long branch history)
- Capacity aliasing (*capacity miss*)
  - Increase PHT size
- Conflict aliasing (*conflict miss*)
  - Change indexing scheme or partition PHT in a clever fashion

Bi-Mode Predictor



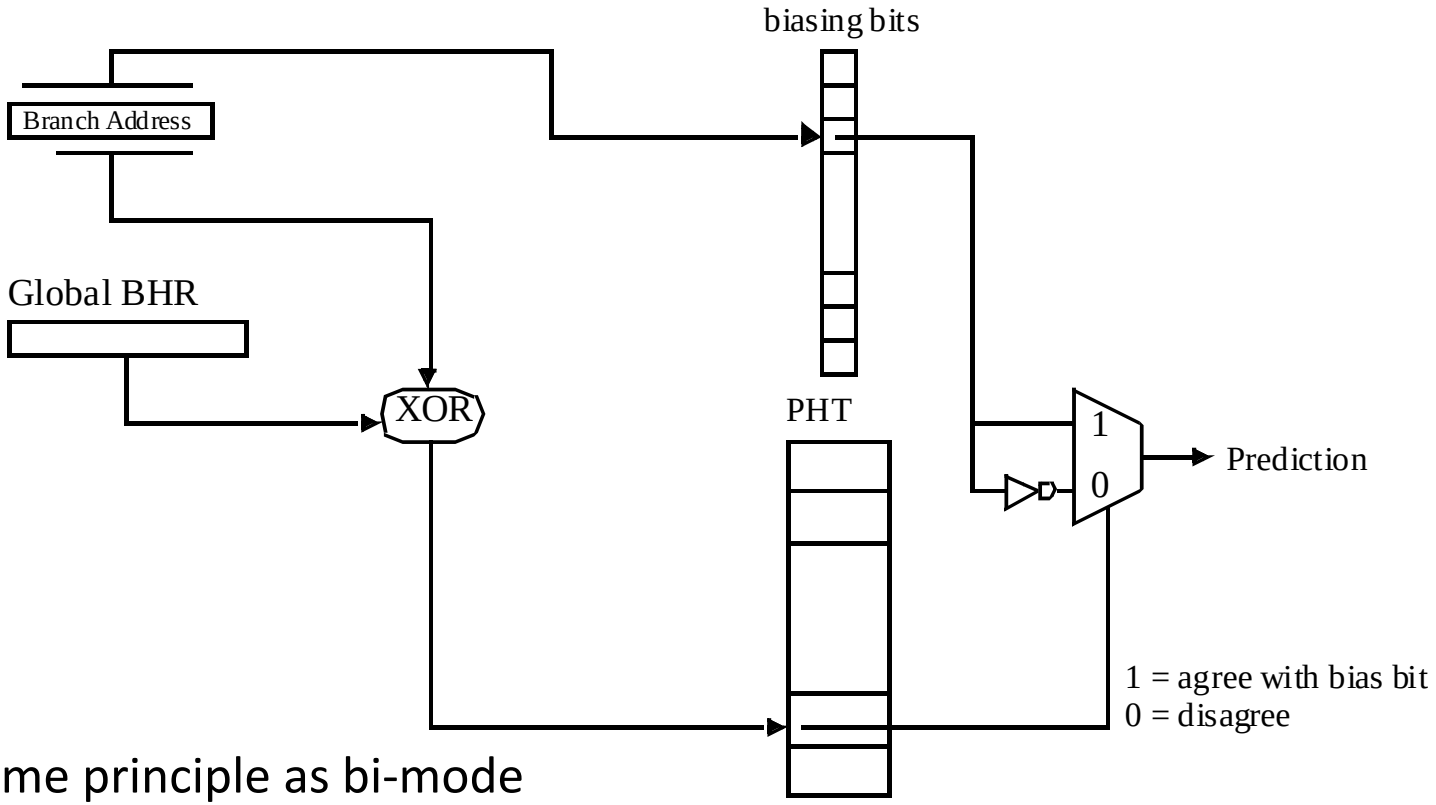
- PHT partitioned into T/NT halves
  - Selector chooses source
- Reduces negative interference, since most entries in PHT_0 tend towards NT, and most entries in PHT_1 tend towards T
- Used by ARM Cortex-A15

gskewed Predictor



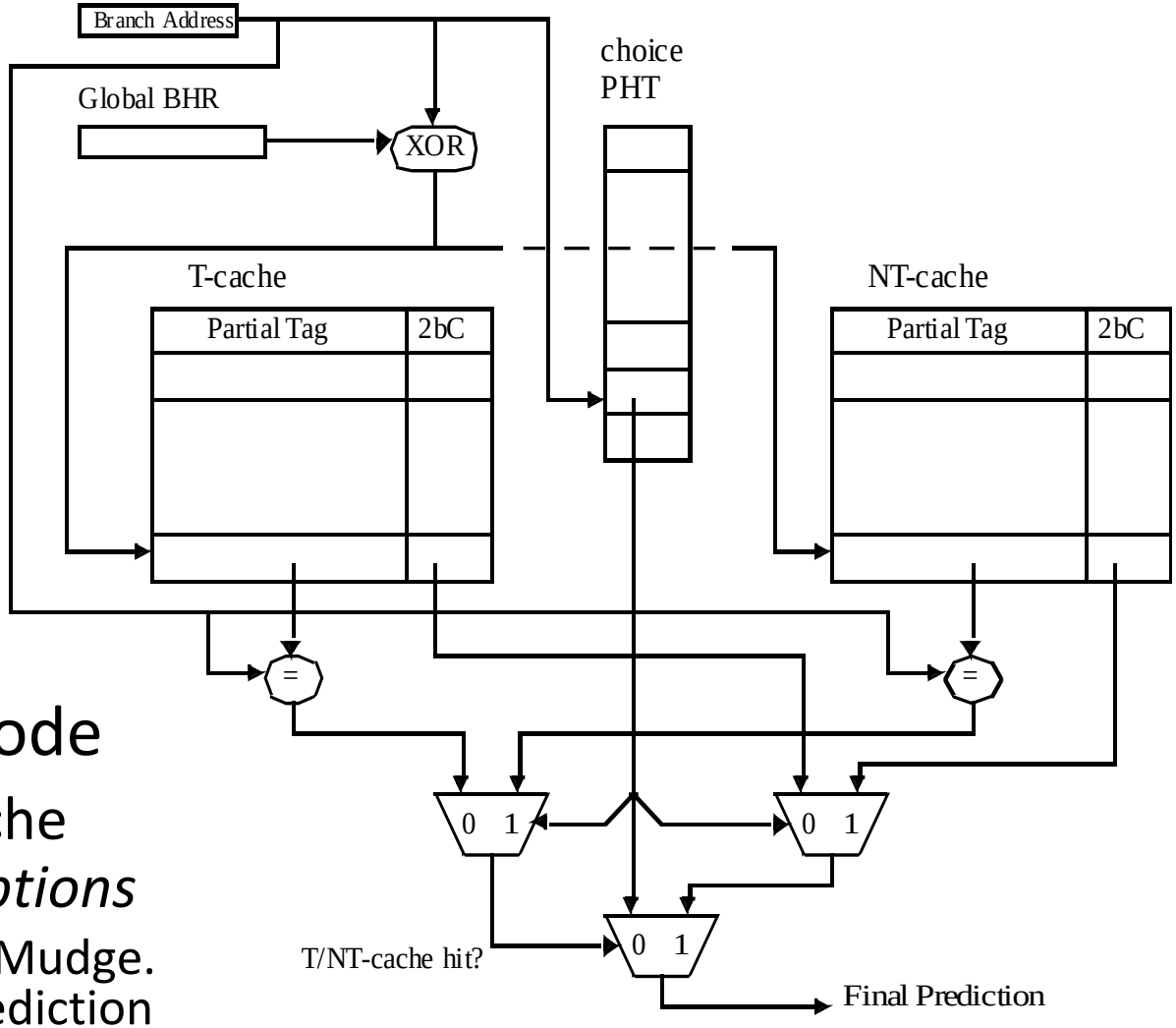
- Multiple PHT banks indexed by different hash functions
  - Conflicting branch pair unlikely to conflict in more than one PHT
- Majority vote determines prediction
- Used in Alpha EV8 (ultimately cancelled)
- P. Michaud, A. Seznec, and R. Uhlig. Trading Conflict and Capacity Aliasing in Conditional Branch Predictors. ISCA-24, June 1997

Agree Predictor



- Same principle as bi-mode
- PHT records whether branch bias matches outcome
  - Exploits 70-80% static predictability
- Used in in HP PA-8700
- E. Sprangle, R. S. Chappell, M. Alsup, and Y. N. Patt. The Agree Predictor: A Mechanism for Reducing Negative Branch History Interference. ISCA-24, June 1997.

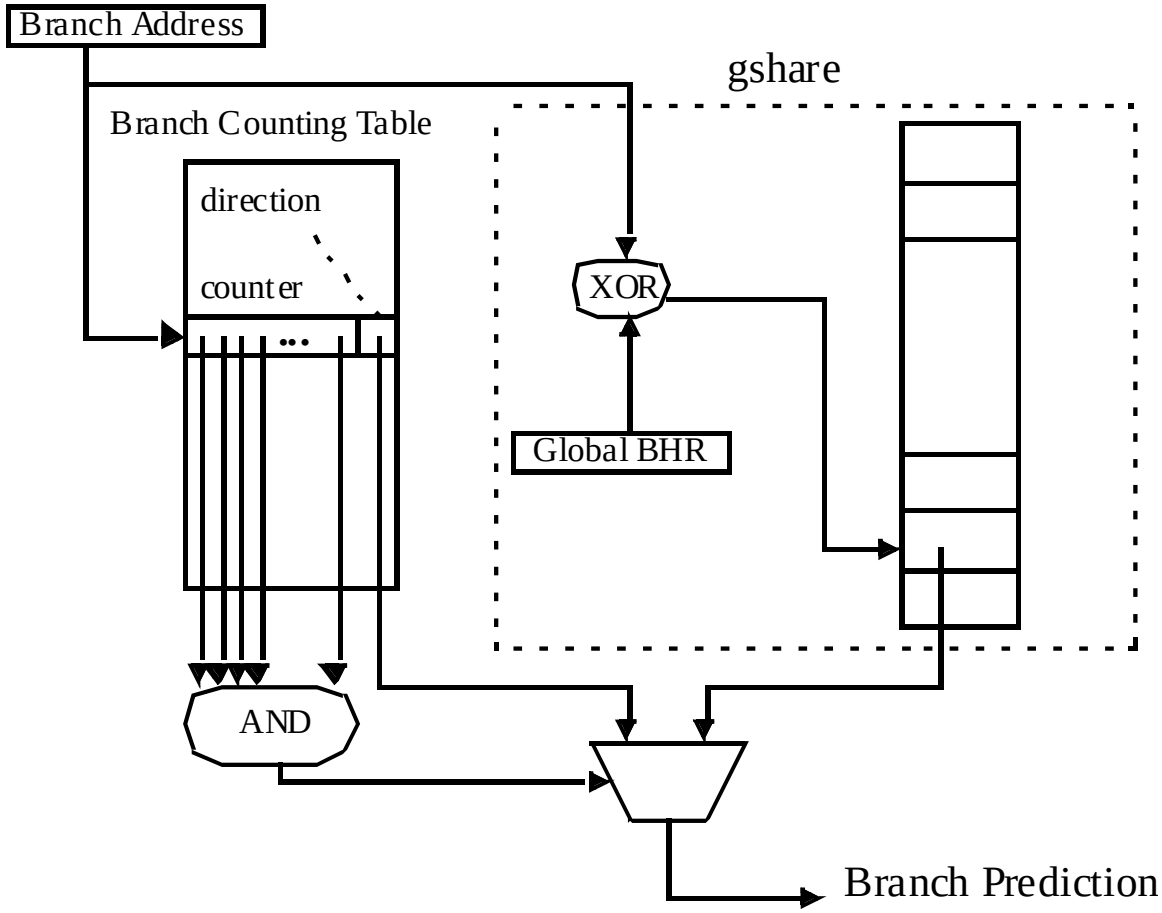
YAGS Predictor



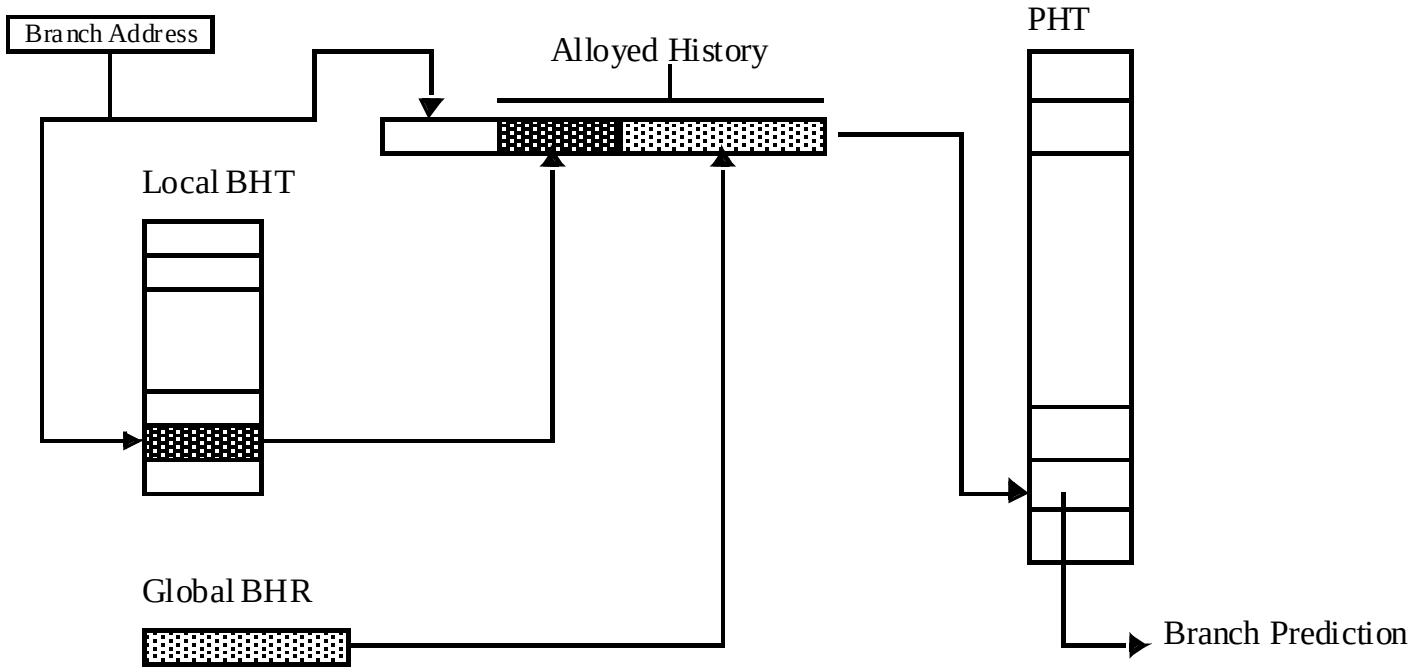
- Based on bi-mode
  - T/NT PHTs cache only the *exceptions*
- A. N. Eden and T. N. Mudge. The YAGS Branch Prediction Scheme. MICRO, Dec 1998.

Branch Filtering

- Highly-biased branches
  - e.g. ‘11111’ history
  - Eliminated from PHT
- P-Y Chang, M. Evers, and Y Patt. Improving Branch Prediction Accuracy by Reducing Pattern History Table Interference. PACT, October 1996.

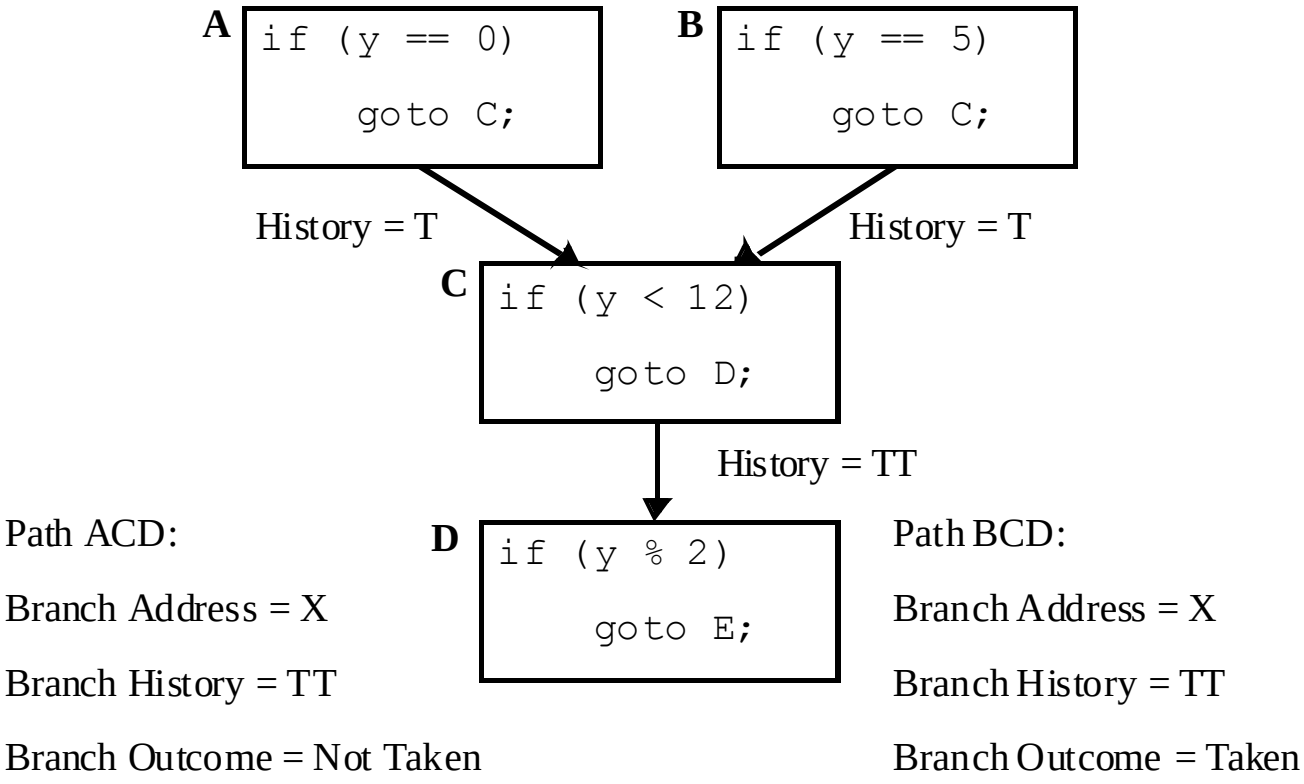


Alloyed-History Predictors



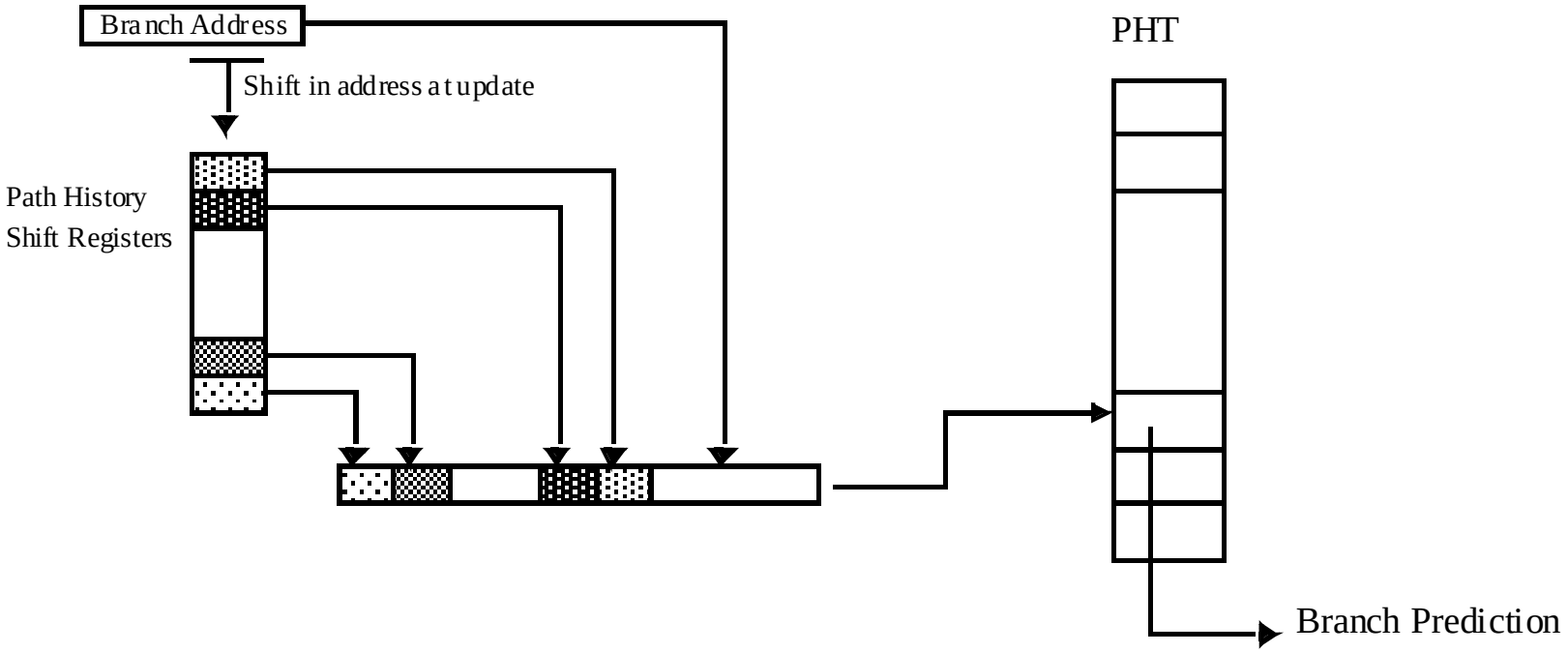
- Local history vs. global history
- Kevin Skadron, Margaret Martonosi, and Douglas W. Clark. Alloyed Global and Local Branch History: A Robust Solution to Wrong-History Mispredictions. International Journal of Parallel Programming, 31(2), April 2003.

Path History



- Sometimes T/NT history is not enough
- Path history (PC values) can help

Path-Based Branch Predictor

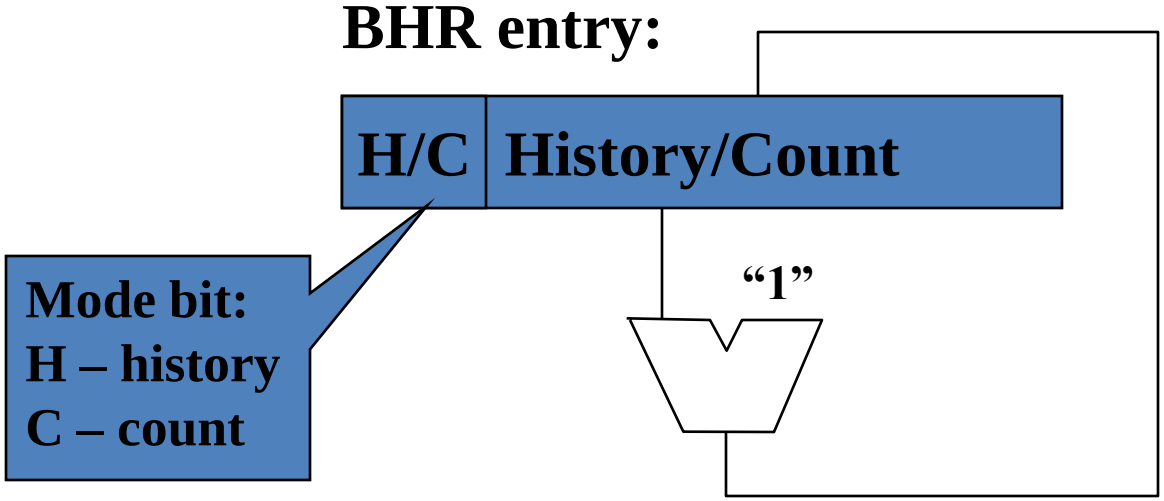


- Ravi Nair. Dynamic Path-Based Branch Correlation. International Symposium on Microarchitecture, pages 15-23, December 1995.

Dynamic History Length

- Branch history length:
  - Some prefer short history (less training time)
  - Some require longer history (complex behavior)
- Vary history length
  - Choose through profile/compile-time hints
  - Or learn dynamically
- References
  - Maria-Dana Tarlescu, Kevin B. Theobald, and Guang R. Gao. Elastic History Buffer: A Low-Cost Method to Improve Branch Prediction Accuracy. ICCD, October 1996.
  - Toni Juan, Sanji Sanjeevan, and Juan J. Navarro. Dynamic History-Length Fitting: A Third Level of Adaptivity for Branch Prediction. ISCA, June 1998.
  - Jared Stark, Marius Evers, and Yale N. Patt. Variable Path Branch Prediction. ACM SIGPLAN Notices, 33(11):170-179, 1998

Loop Count Predictors



- To predict last loop iteration's NT branch:
  - Must have length(BHR) > loop count
  - Not feasible for large loop counts
- Instead, BHR has mode bit
  - Once history == '111…11' or '000…00' switch to count mode
  - Now n^{th} entry in PHT trains to NT and predicts n^{th} iteration as last one
  - Now length(BHR) > $\log_2$(loop count) is sufficient
- Used in Intel Pentium M/Core Duo/ Core 2 Duo

Understanding Advanced Predictors

- Four types of history
  - Local (bimodal) history (Smith predictor)
    - Table of counters summarizes local history
    - Simple, but only effective for biased branches
  - Local outcome history (correlate with self)
    - Shift register of individual branch outcomes
    - Separate counter for each outcome history (M-F vs Sat/Sun)
  - Global outcome history (correlate with others)
    - Shift register of recent branch outcomes
    - Separate counter for each outcome history
  - Path history (overcomes CFG convergence aliasing)
    - Shift register of recent (partial) block addresses
    - Can differentiate similar global outcome histories
- Can combine or “alloy” histories in many ways

Understanding Advanced Predictors

- History length
  - Short history—lower training cost
  - Long history—captures macro-level behavior
  - Variable history length predictors
- Really long history (long loops)
  - Loop count predictors
  - Fourier transform into frequency domain
    - Kampe et. al, “The FAB Predictor…”, HPCA 2002
- Limited capacity & interference
  - Constructive vs. destructive
  - Bi-mode, gskewed, agree, YAGS
  - Sec. 9.3.2 provides good overview

Perceptron Branch Prediction

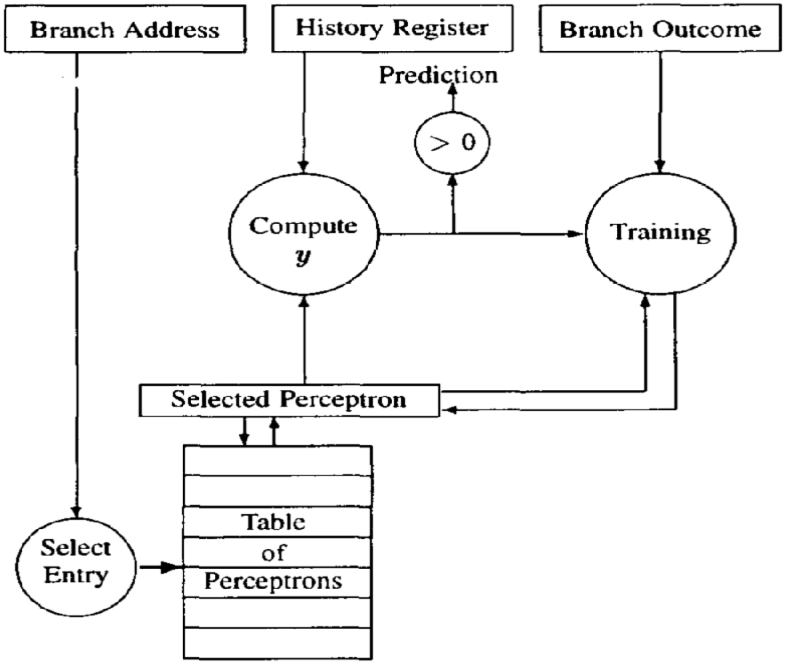
$$y = w_0 + \sum_{i=1}^{n} x_i w_i$$

[Jimenez, Lin HPCA 2001]

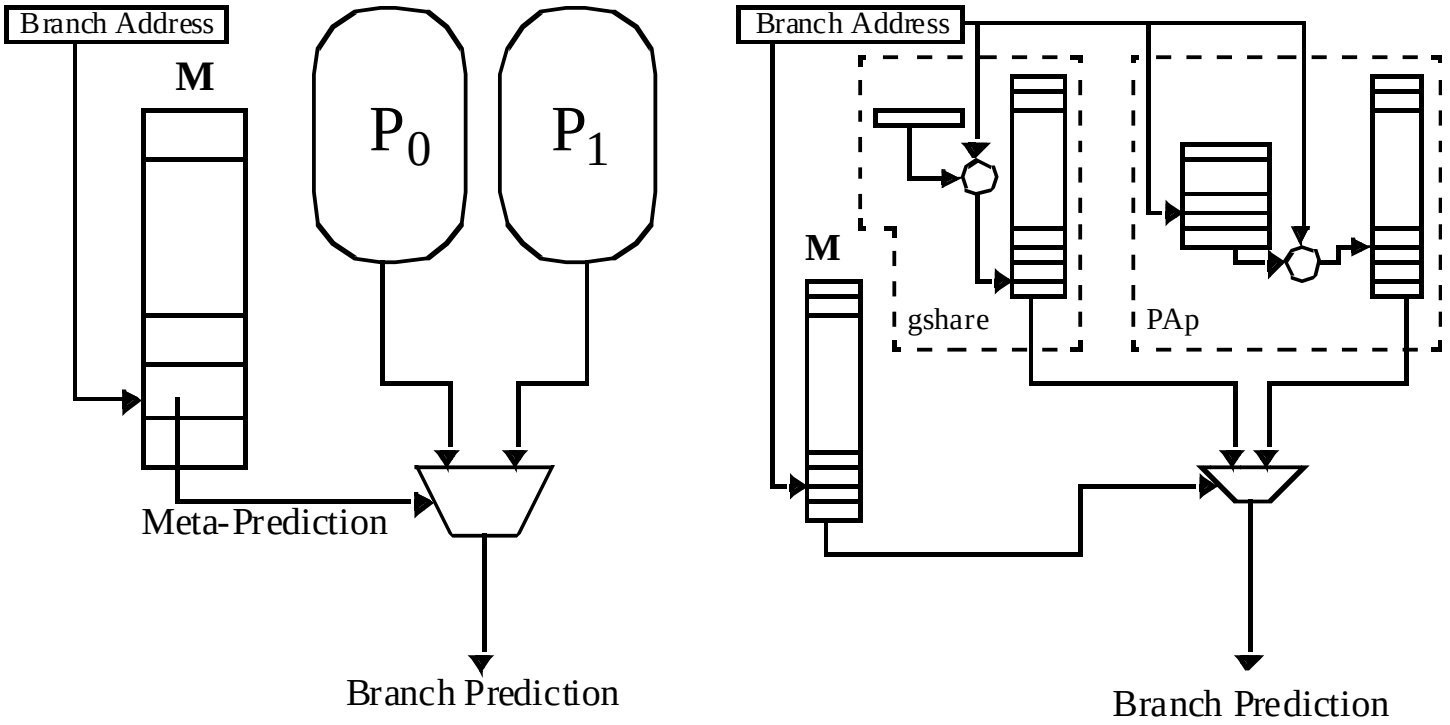
- Perceptron
  - Basis in AI concept [1962]
  - Computes boolean result based on multiple weighted inputs
- Adapted for branch prediction
  - x_i from branch history (1 T, -1 NT)
  - w_i incremented whenever branch outcome matches xi
  - Finds correlation between current branch and any subset of prior branches

Perceptrons - Implementation

- Complex dot product must be computed for every prediction
  - Too slow
- Arithmetic tricks, pipelining:
  - Daniel A. Jimenez and Calvin Lin. Neural methods for dynamic branch prediction. ACM Transactions on Computer Systems, 20(4):369–397, November 2002.
  - Analog circuit implementation also possible
    - Amant, Jimenez, Burger, MICRO 2008
- Key insight:
  - Not all branches in history are important (correlate)
  - Perceptron weights learn this

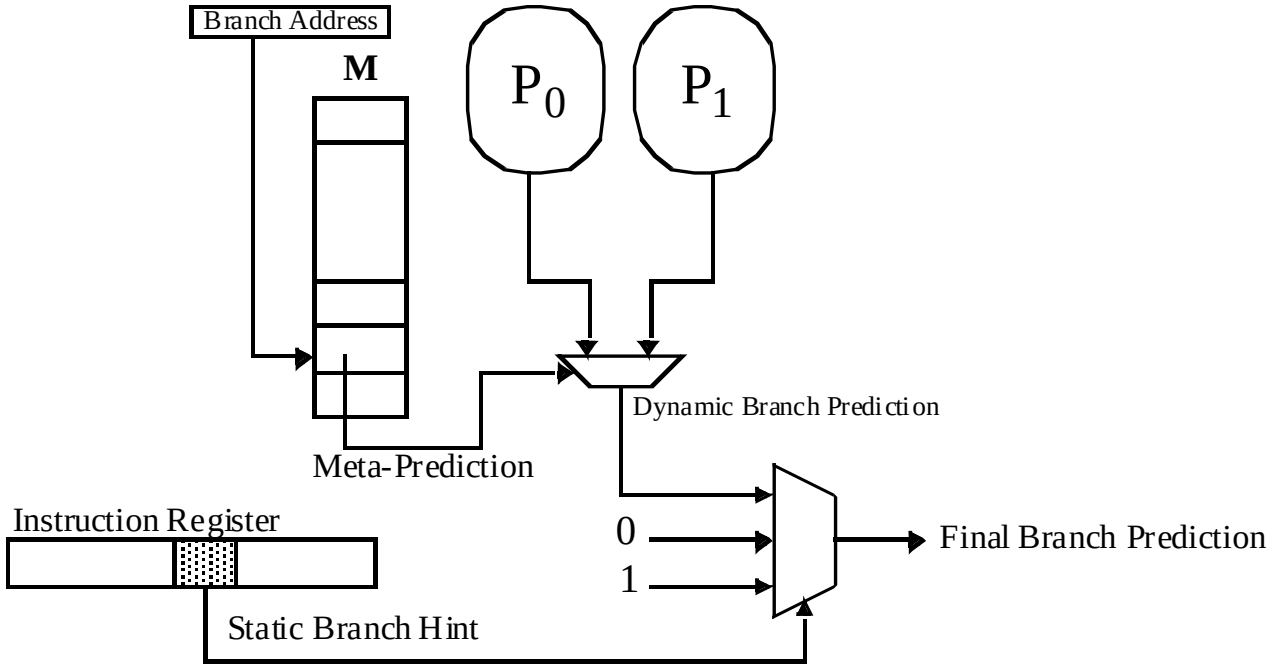


Combining or Hybrid Predictors



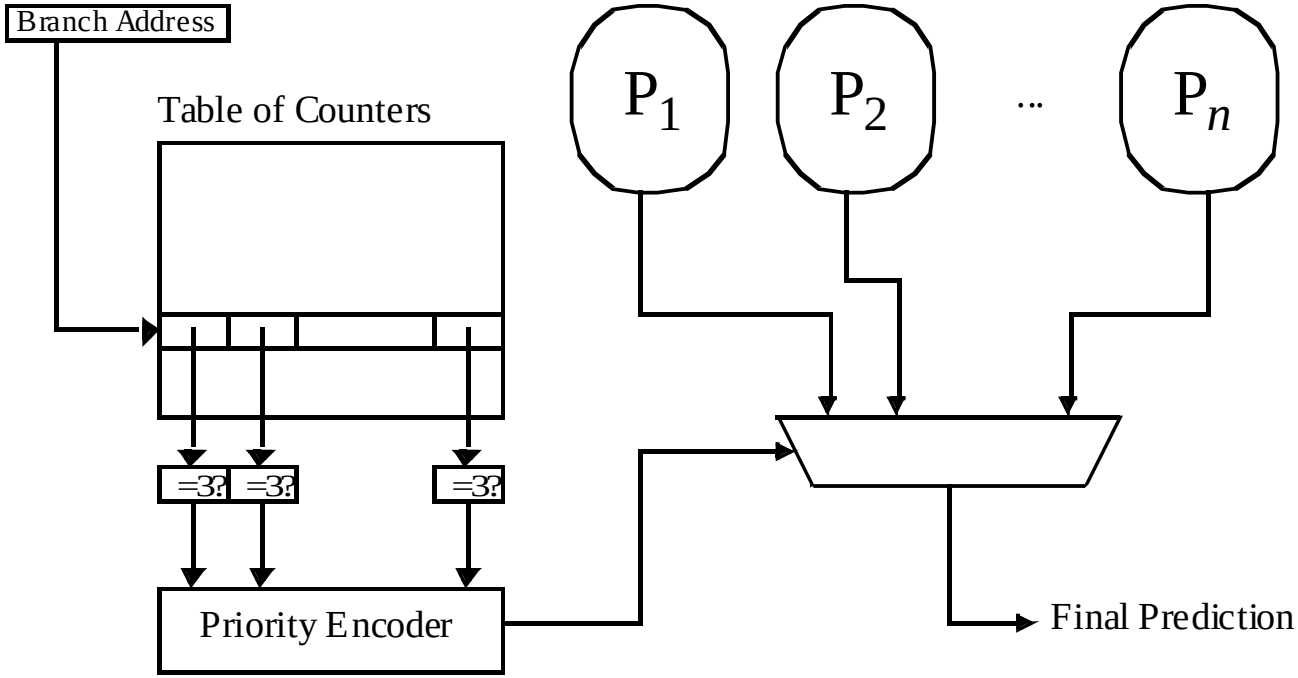
- Select “best” history
- Reduce interference w/partial updates
- Scott McFarling. Combining Branch Predictors. TN-36, Digital Equipment Corporation Western Research Laboratory, June 1993.

Branch Classification



- Static (profile-based) branch hints select which prediction to use
  - Static T/Static NT/Dynamic
  - PowerPC y-bit overrides static BTFN
- P-Y Chang, E Hao, TY Yeh, and Y Patt. Branch Classification: a New Mechanism for Improving Branch Predictor Performance. MICRO, Nov. 1994.
- D Grunwald, D Lindsay, and B Zorn. Static Methods in Hybrid Branch Prediction. PACT, October 1998

Multi-Hybrid Predictor



- Generalizes selector to choose from > 2 predictors
- Marius Evers, Po-Yung Chang, and Yale N. Patt. Using Hybrid Branch Predictors to Improve Branch Prediction Accuracy in the Presence of Context Switches. International Symposium on Computer Architecture, pages 3-11, May 1996.

Overriding Predictors

- Different types of history
  - E.g. Bimodal, Local, Global (BLG)
- Different history lengths (up to hundreds of branches)
- How to choose?
  - Metapredictor/selector? Expensive, slow to train
- Tag match with most sophisticated predictor entry
  - Parallel tag check with B, L, G, long-history G
  - Choose most sophisticated prediction
  - Fancy predictors only updated when simple ones fail

Current State of the Art

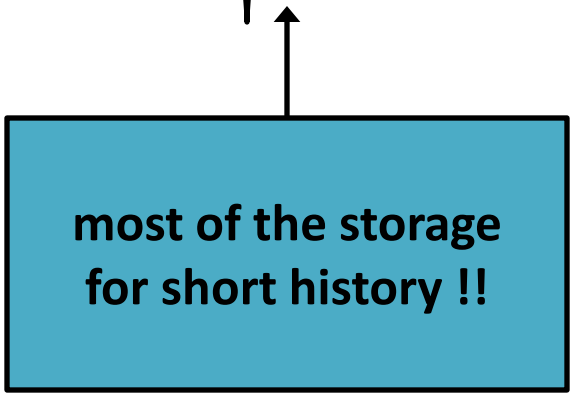
- Key concepts
  - Different history type (B,L,G)
  - Variable history length
    - Some branches prefer short, others long
    - Use geometric series [Seznec, CBP-1, O-GEHL]
  - Caching only exceptions
    - Partial tag match (YAGS)
  - Confidence estimation [Jacobson et al, MICRO 1996]
- Tagged Geometric History Length (TAGE)
  - A. Seznec, P. Michaud, "A case for (partially) tagged Geometric History Length Branch Prediction", Journal of Instruction Level Parallelism , Feb. 2006

TAGE Predictor

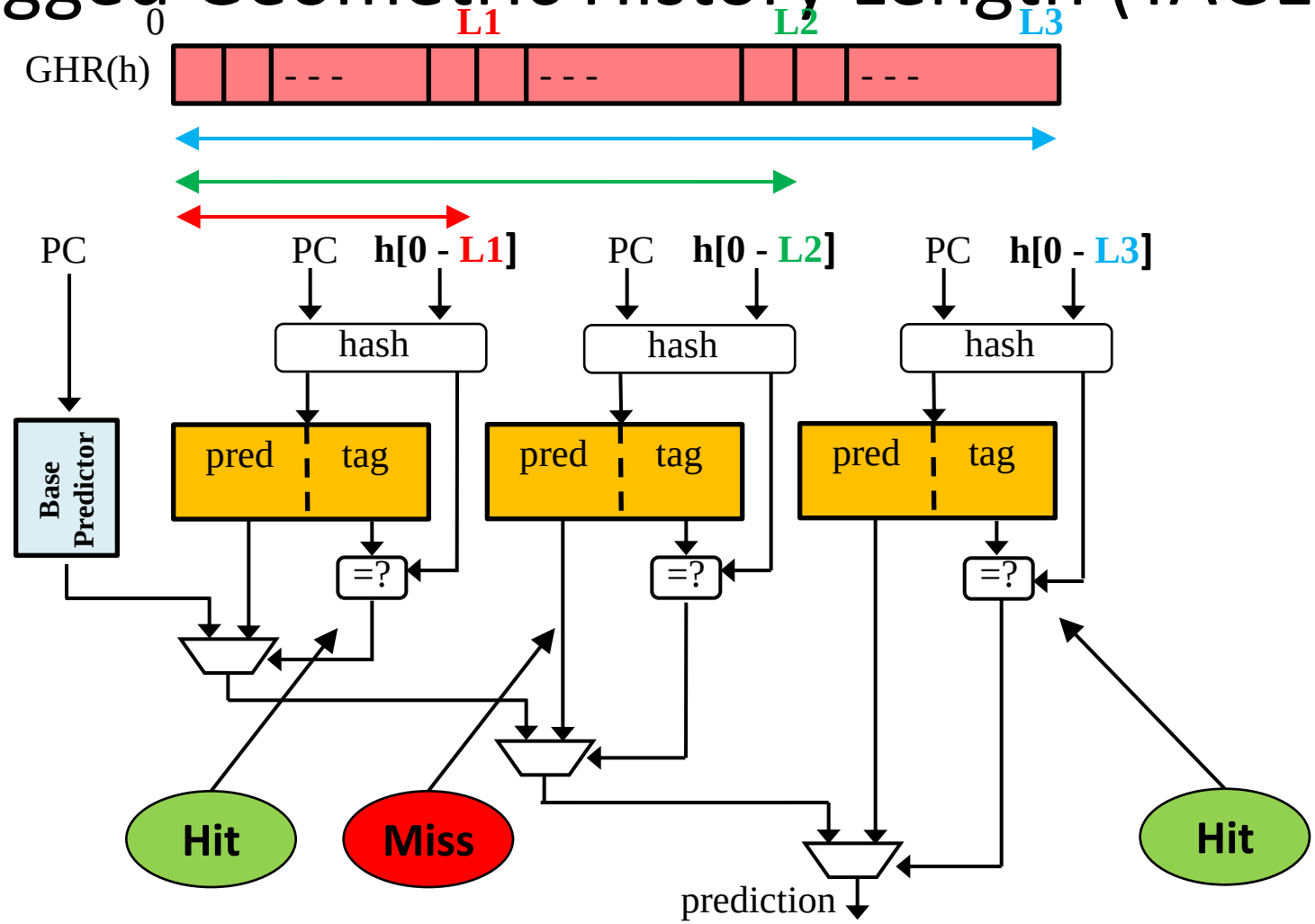
- Multiple tagged tables, use different global history lengths

- Set of history lengths forms a geometric series

{0, 2, 4, 8, 16, 32, 64, 128, 256, …, 2048}



Tagged Geometric History Length (TAGE)



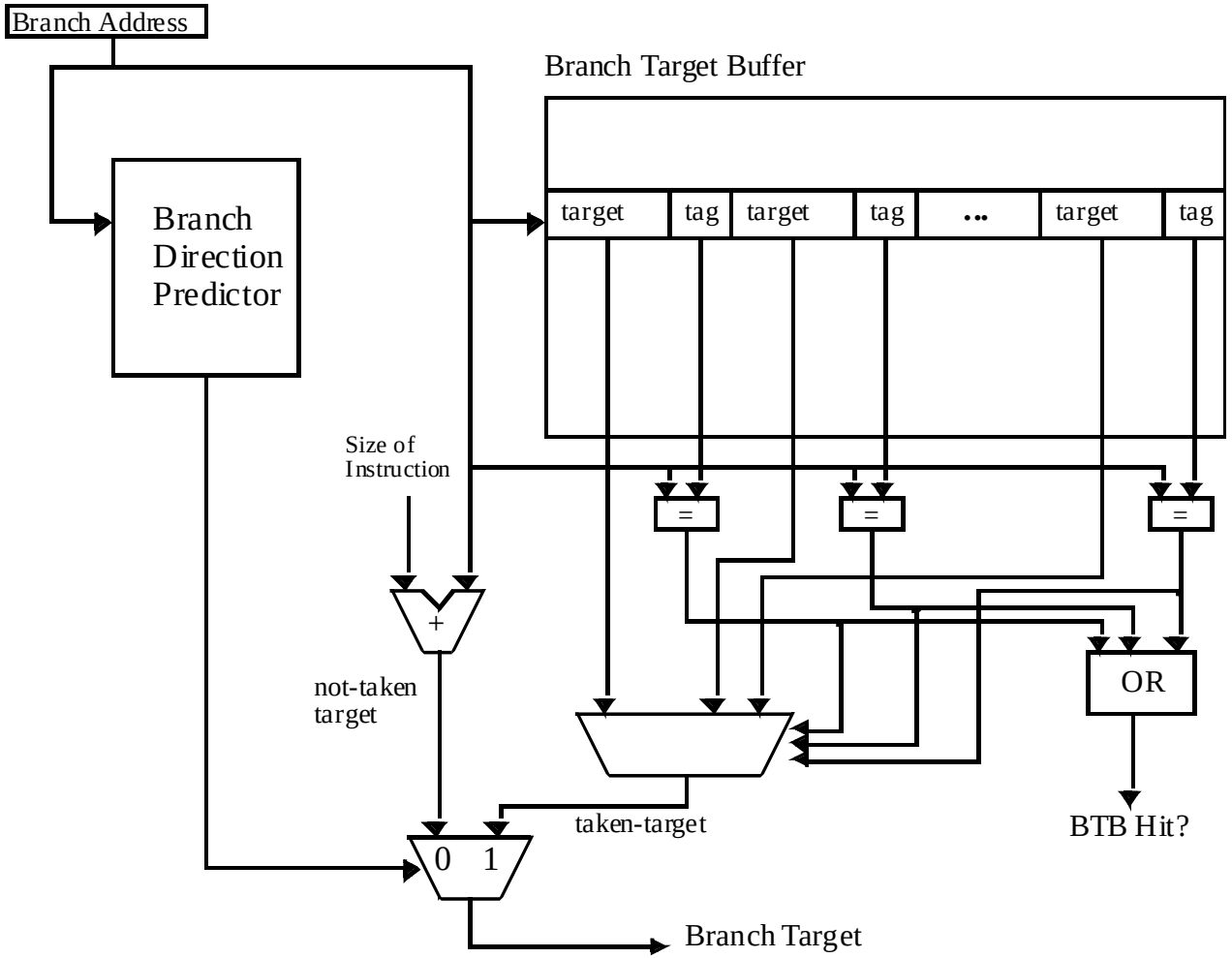
- Longest matching table provides the prediction, subject to branch confidence

TAGE

- Minor tweaks to basic concept still win CBP-x
- State of the art, but…
  - Not yet implemented in a practical design
  - Very expensive hardware
  - Very energy-intensive (parallel lookups)
  - Complex update rules

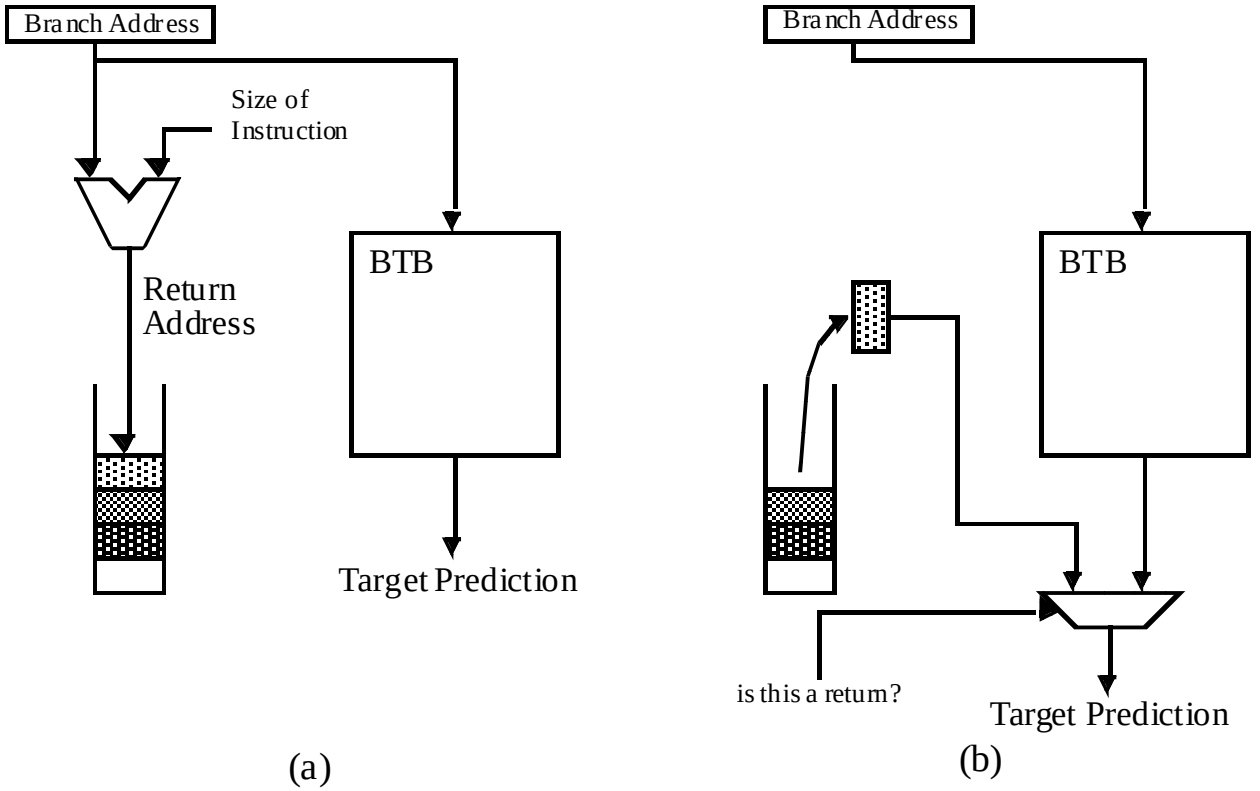
- Real opportunity exists for improvement

Branch Target Prediction



- Partial tags sufficient in BTB

Return Address Stack



- Speculative update causes headaches
  - On each predicted branch, checkpoint head/tail
  - Further, checkpoint stack contents since speculative pop/push sequence is destructive
  - Conditional call/return causes more headaches

Indirect Branches

- Tagged target cache
  - Chang et. al, Target Prediction for Indirect Jumps, ISCA 1997

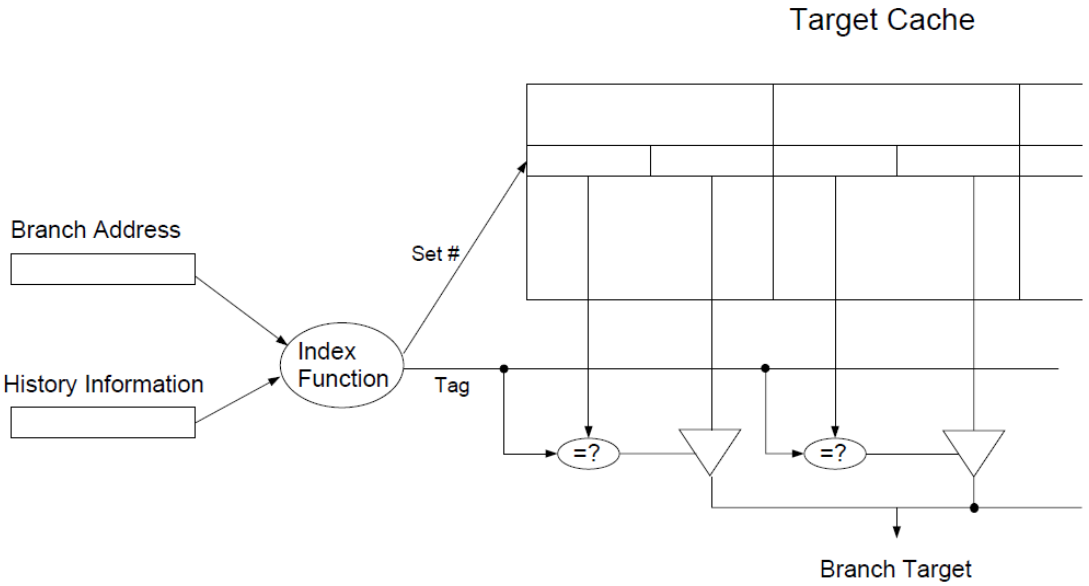


Figure 11: Structure of a Tagged Target Cache

Indirect Branches

- ITTAGE proposed in same 2006 paper as TAGE
  - A. Seznec, P. Michaud, "A case for (partially) tagged Geometric History Length Branch Prediction", Journal of Instruction Level Parallelism , Feb. 2006

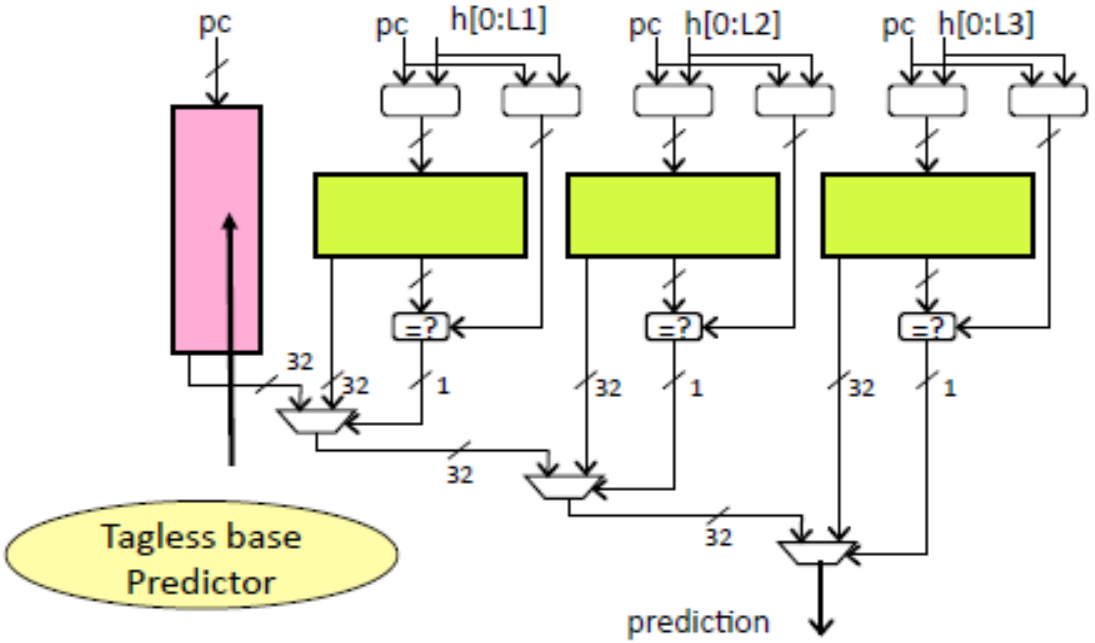
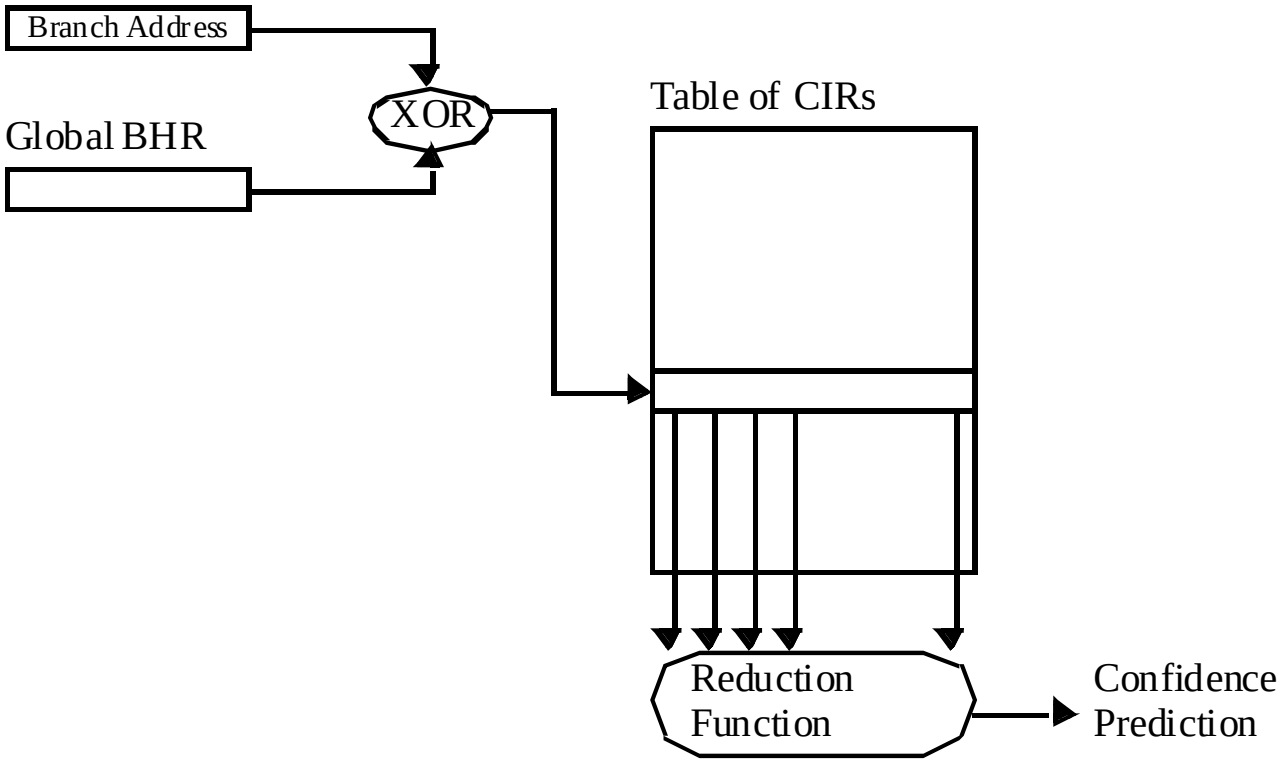


Figure 1: The Indirect Target TAgged GEometric length, ITTAGE, predictor

Indirect Branches

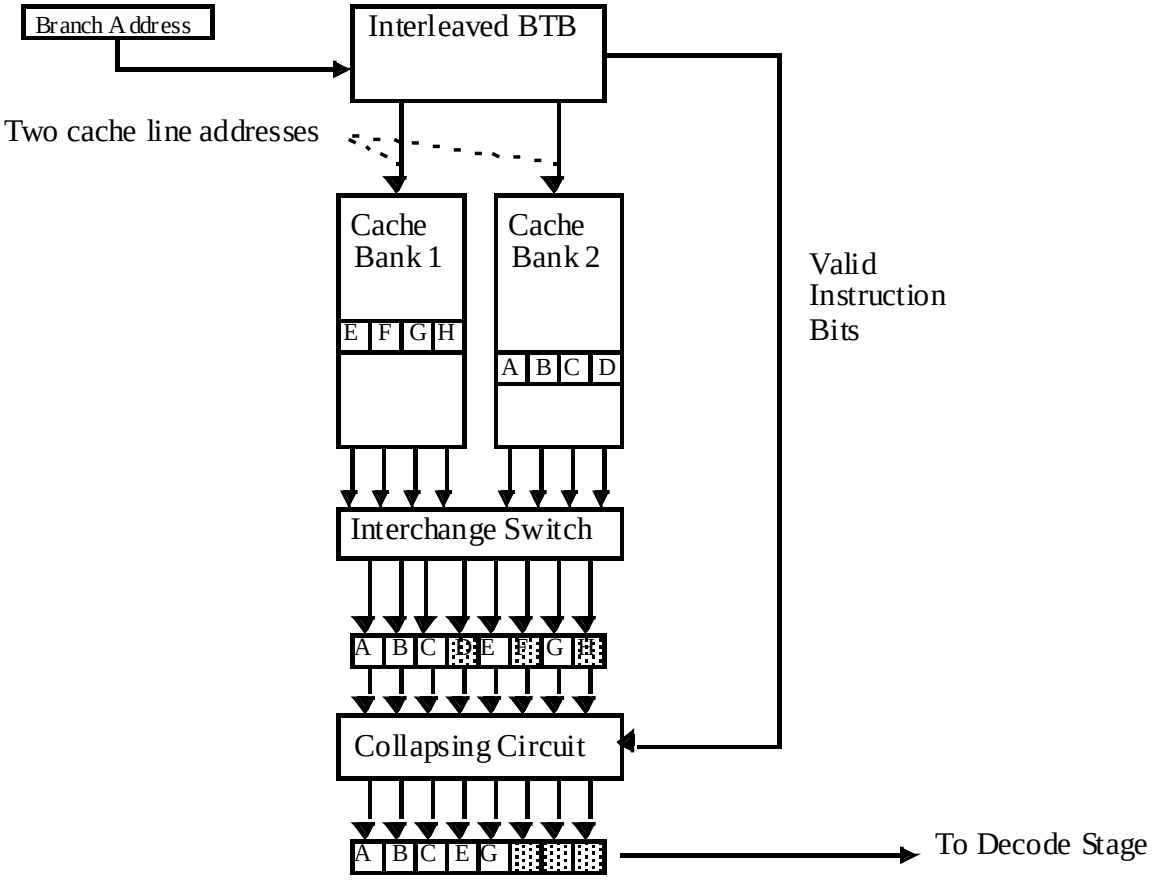
- CPB-3 had an indirect prediction track
  - #1: A. Seznec, *A 64-Kbytes ITTAGE indirect branch predictor,* MPPKI 34.1
  - #2: Y. Ishii, T. Sawada, K. Kuroyanagi, M. Inaba, K. Hiraki*, Bimode Cascading: Adaptive Rehashing for ITTAGE Indirect Branch Predictor,* MPPKI 37.0
  - #3: N. Bhansali, C. Panirwala, H. Zhou, *Exploring Correlation for Indirect Branch Prediction,* MPPKI 51.6
  - #4: Daniel A. Jimenez, *SNIP: Scaled Neural Indirect Predictor,* MPPKI *52.9*

Branch Confidence Estimation



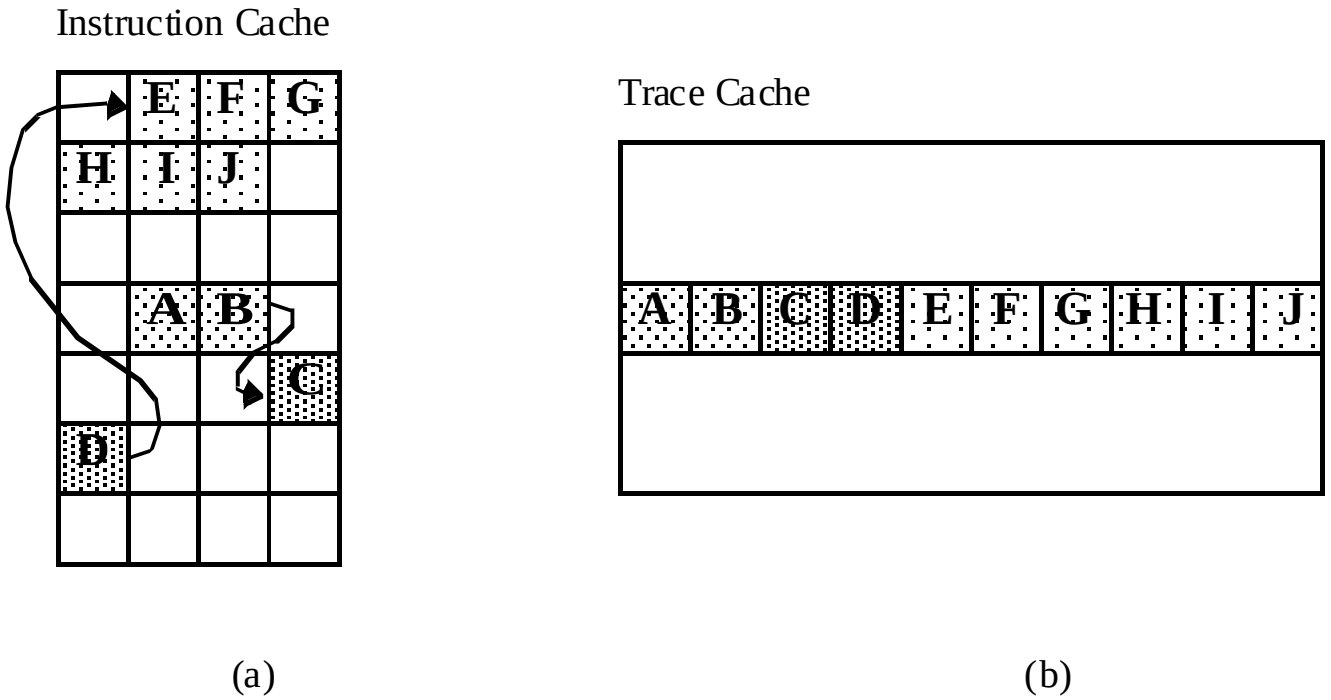
- Limit speculation (energy), reverse predictions, guide fetch for multithreaded processors, choose best prediction
- Q Jacobson, E Rotenberg, and JE Smith. Assigning Confidence to Conditional Branch Predictions. MICRO, December 1996.

High-Bandwidth Fetch: Collapsing Buffer



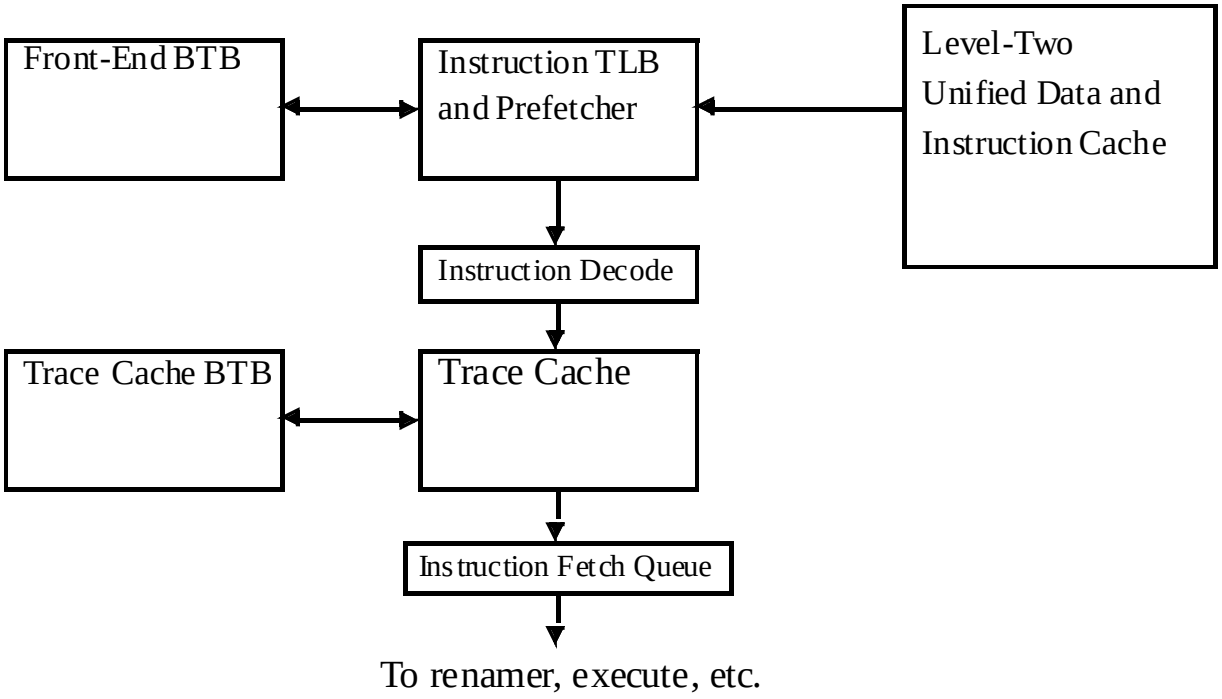
- Fetch from two cache blocks, rotate, collapse past taken branches
- Thomas M. Conte, Kishore N. Menezes, Patrick M. Mills and Burzin A. Patel. Optimization of Instruction Fetch Mechanisms for High Issue Rates. International Symposium on Computer Architecture, June 1995.

High-Bandwidth Fetch: Trace Cache



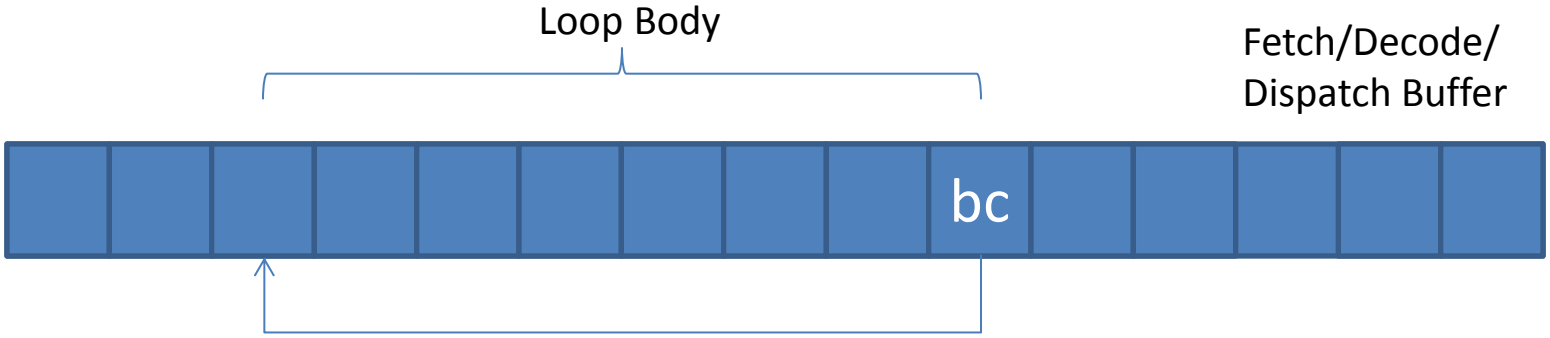
- Fold out taken branches by *tracing* instructions as they commit into a *fill buffer*
- Eric Rotenberg, S. Bennett, and James E. Smith. Trace Cache: A Low Latency Approach to High Bandwidth Instruction Fetching. MICRO, December 1996.

Intel Pentium 4 Trace Cache



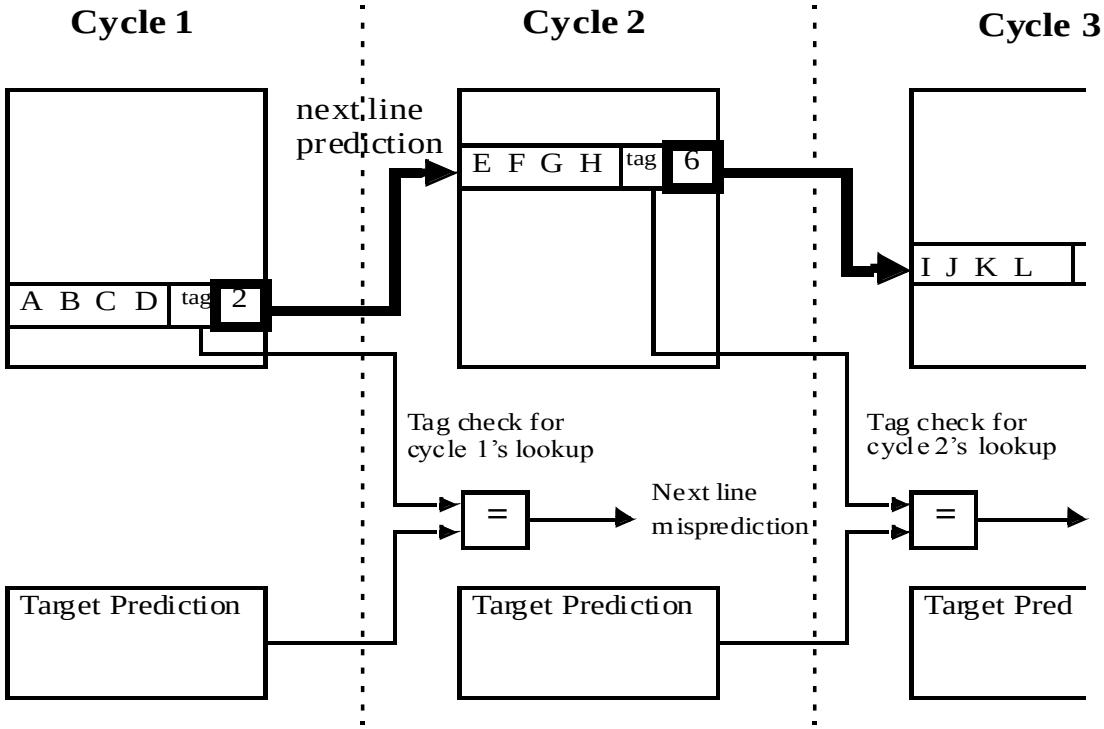
- No first-level instruction cache: trace cache only
- Trace cache BTB identifies next trace
- Miss leads to fetch from level two cache
- Trace cache instructions are decoded (uops)
- Cache capacity 12k uops
  - Overwhelmed for database applications
  - Serial decoder becomes performance bottleneck

High-Bandwidth Fetch: Loop Buffers



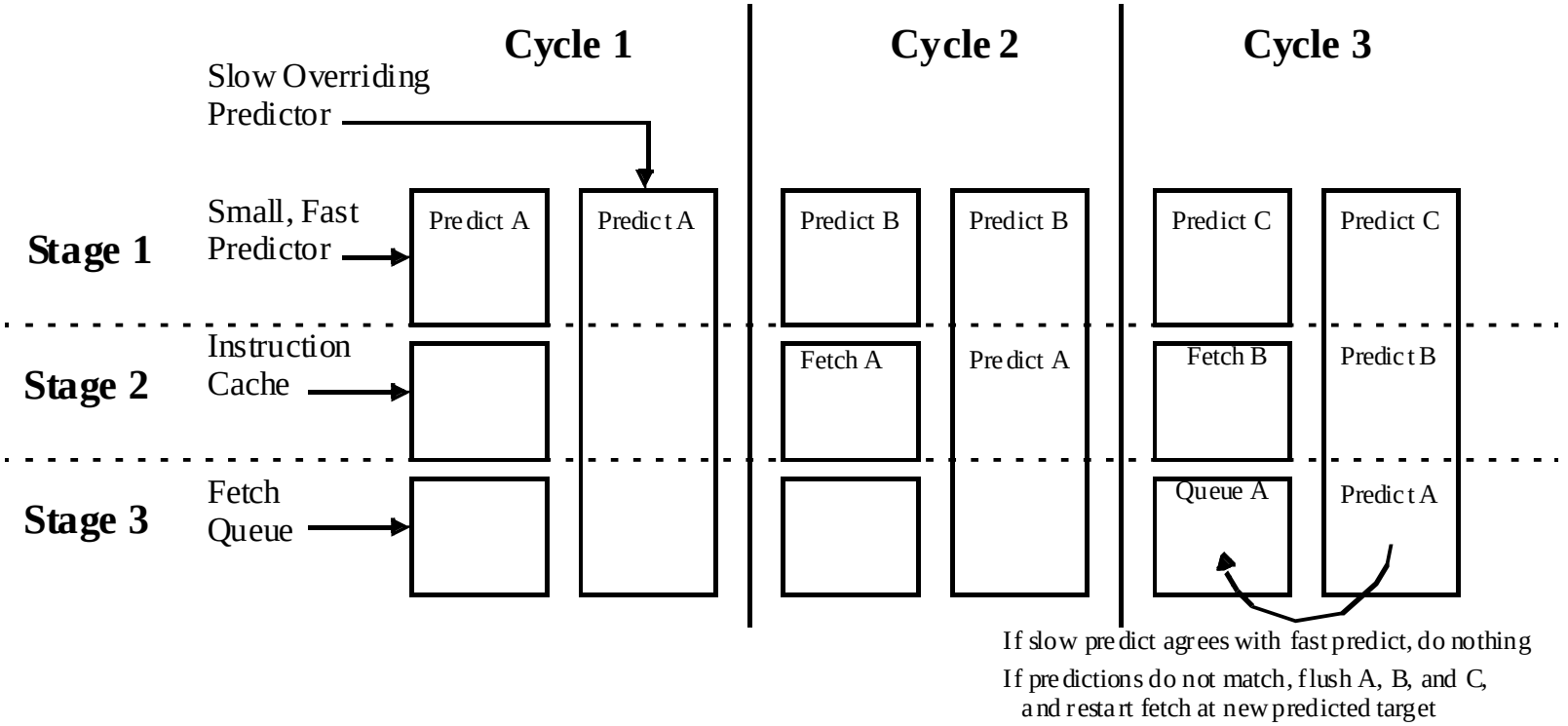
- History: AMD29K Branch Target Cache
  - Don't cache the target address; cache 4 instructions from the target itself
  - Avoid accessing I$ for first fetch group following a taken branch
  - If loop body is <= 4 instructions, effectively a loop cache
  - Room for 32/64 branch targets
- Also common in DSP designs, under s/w control (e.g. Lucent)
- Introduced in Intel Merom (Core 2 Duo)
  - Fetch buffer detects short backward branches, inhibits refetch from I$
- Intel Nehalem (Core i7)
  - Moved loop buffer after decoders: contains uops
- Intel Sandybridge
  - General-purpose uop cache (not just loops)
  - 1.5K capacity

High Frequency: Next-line Prediction



- Embed next fetch address in instruction cache
  - Enables high-frequency back-to-back fetch
- Brad Calder and Dirk Grunwald. Next Cache Line and Set Prediction. International Symposium on Computer Architecture, pages 287-296, June 1995.

High Frequency: Overriding Predictors



- Simple, fast predictor turns around every cycle
- Smarter, slower predictor can override
- Widely used: PowerPC 604, 620, Alpha 21264

Advanced Branch Prediction Summary

- Control Flow Speculation
  - Branch Speculation
  - Mis-speculation Recovery
- Branch Direction Prediction
  - Static Prediction
  - Dynamic Prediction
  - Hybrid Prediction
  - TAGE
- Branch Target Prediction
- High-bandwidth Fetch
- High-Frequency Fetch