18-747 Lecture 9: Advanced Branch Prediction

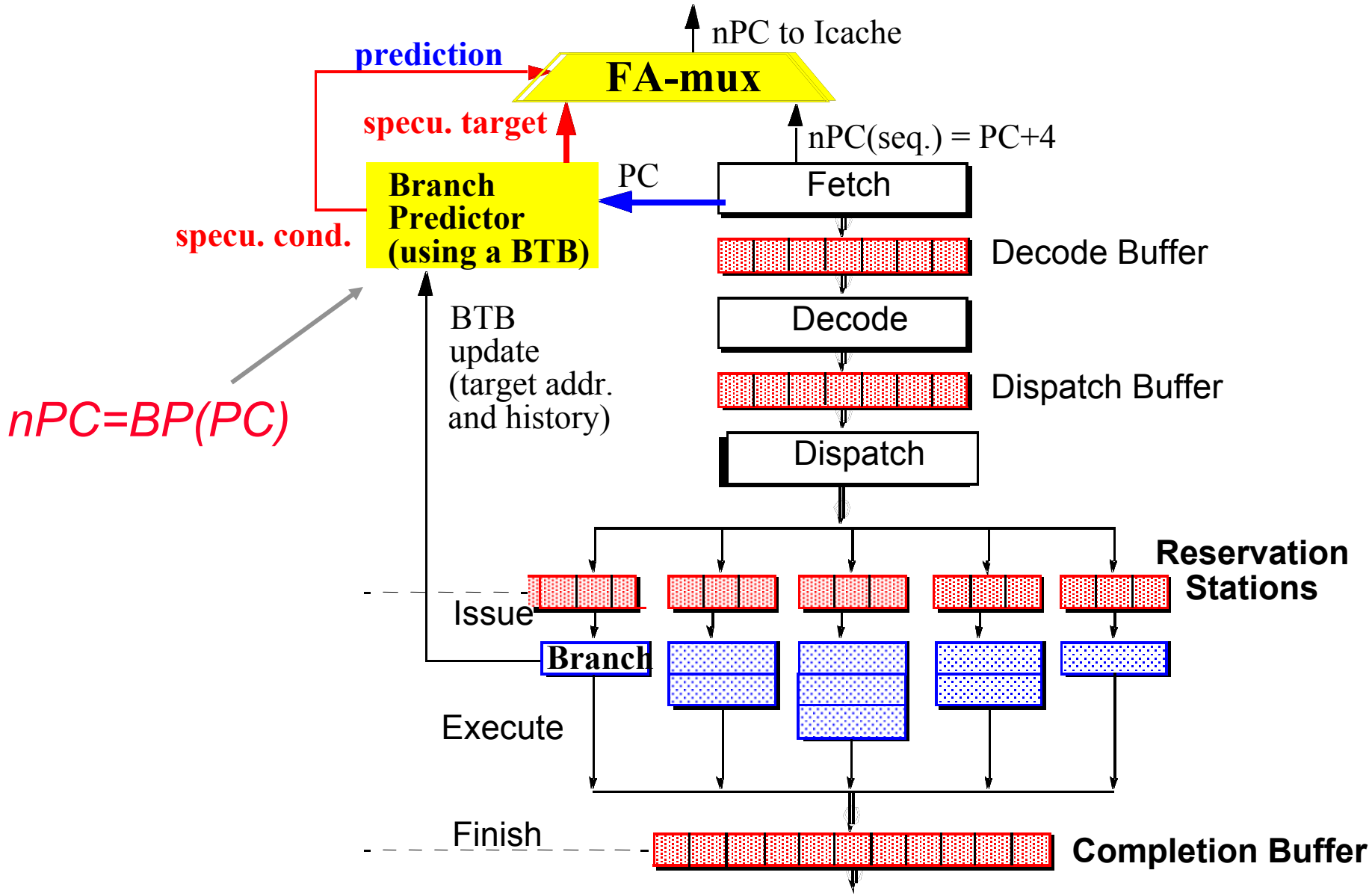
James C. Hoe
Dept of ECE, CMU
September 26, 2001

Reading Assignments: S&L Ch3 82-107, MJ Ch4 and Ch5, McFarling paper

Announcements: Exam 1 on October 15th
Condor Job Queue

Handouts: *Practice Exam 1*

Branch Instruction Speculation



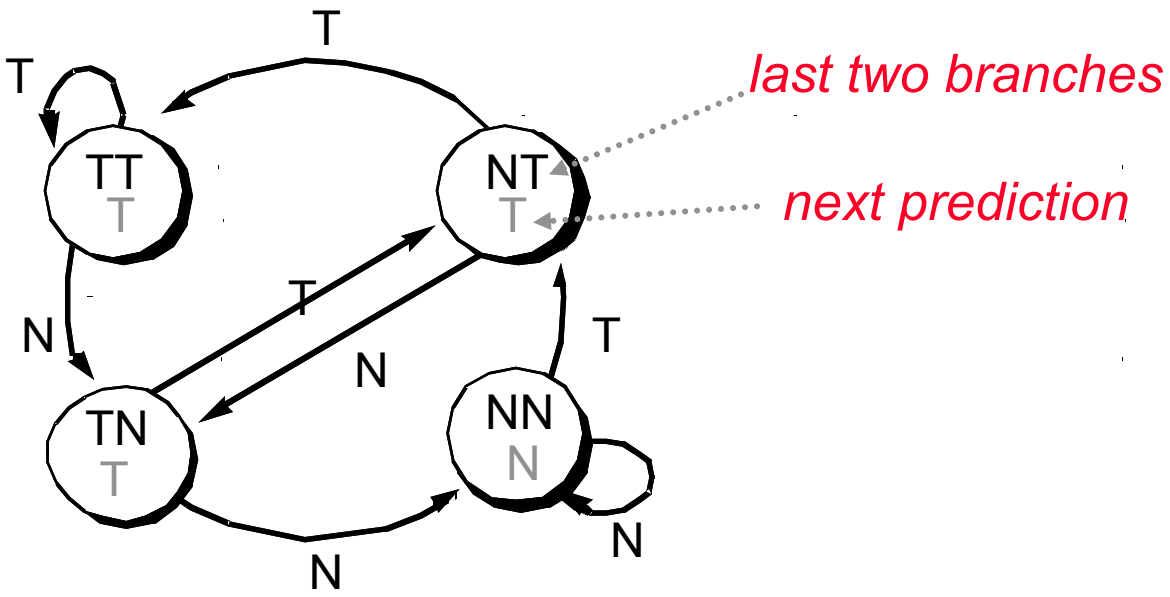
Branch Target Buffer (BTB)

- A small "cache-like" memory in the instruction fetch stage

- Remembers previously executed branches, their addresses, information to aid prediction, and most recent target addresses
- Instruction fetch stage compares current PC against those in BTB to "guess" nPC
  - *If* matched *then* prediction is made *else* nPC=PC+4
  - *If* predict taken *then* nPC=target address in BTB *else* nPC=PC+4
- When branch is actually resolved, BTB is updated

Example Prediction Algorithm

- Prediction accuracy approaches maximum with as few as 2 preceding branch occurrences used as history



Branch Prediction Function

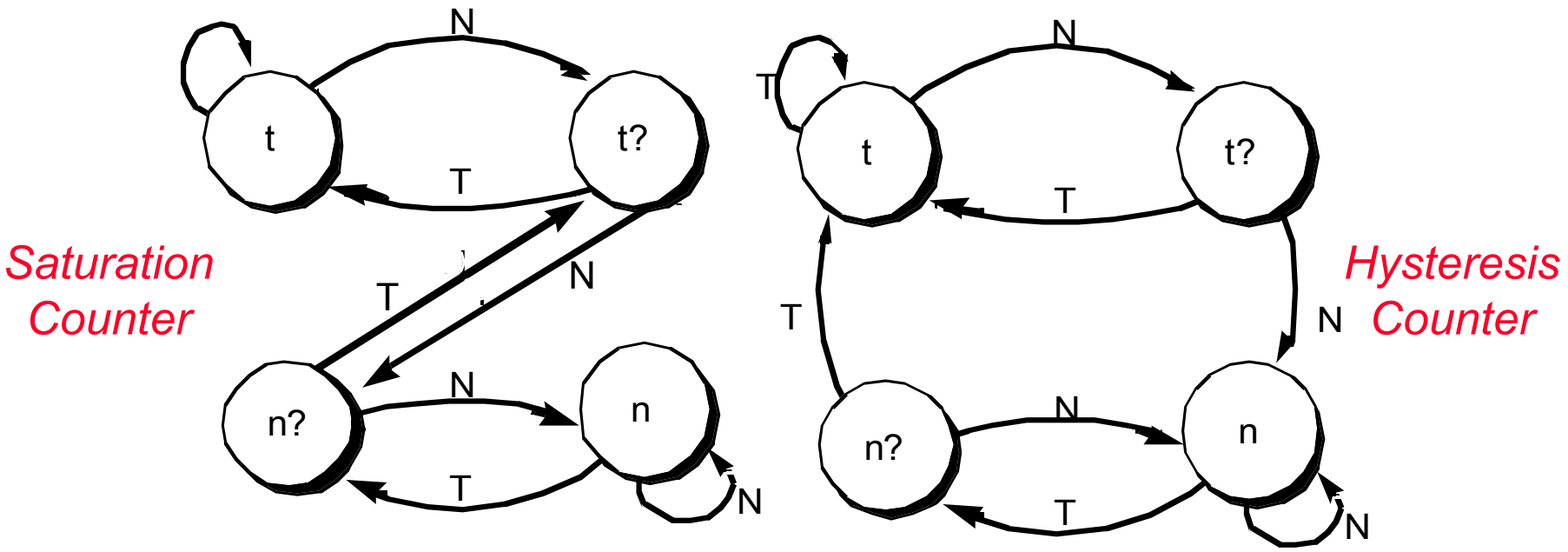
- Based on opcode only (%)

IBM1	IBM2	IBM3	IBM4	DEC	CDC
66	69	71	55	80	78

- Based on history of branch
  - Branch prediction function prediction F (X1, X2, ...)
  - Use up to 5 previous branches for history (%)

	IBM1	IBM2	IBM3	IBM4	DEC	CDC
0	64.1	64.4	70.4	54.0	73.8	77.8
1	91.9	95.2	86.6	79.7	96.5	82.3
2	93.3	96.5	90.8	83.4	97.5	90.6
3	93.7	96.7	91.2	83.5	97.7	93.5
4	94.5	97.0	92.0	83.7	98.1	95.3
5	94.7	97.1	92.2	83.9	98.2	95.7

Other Prediction Algorithms

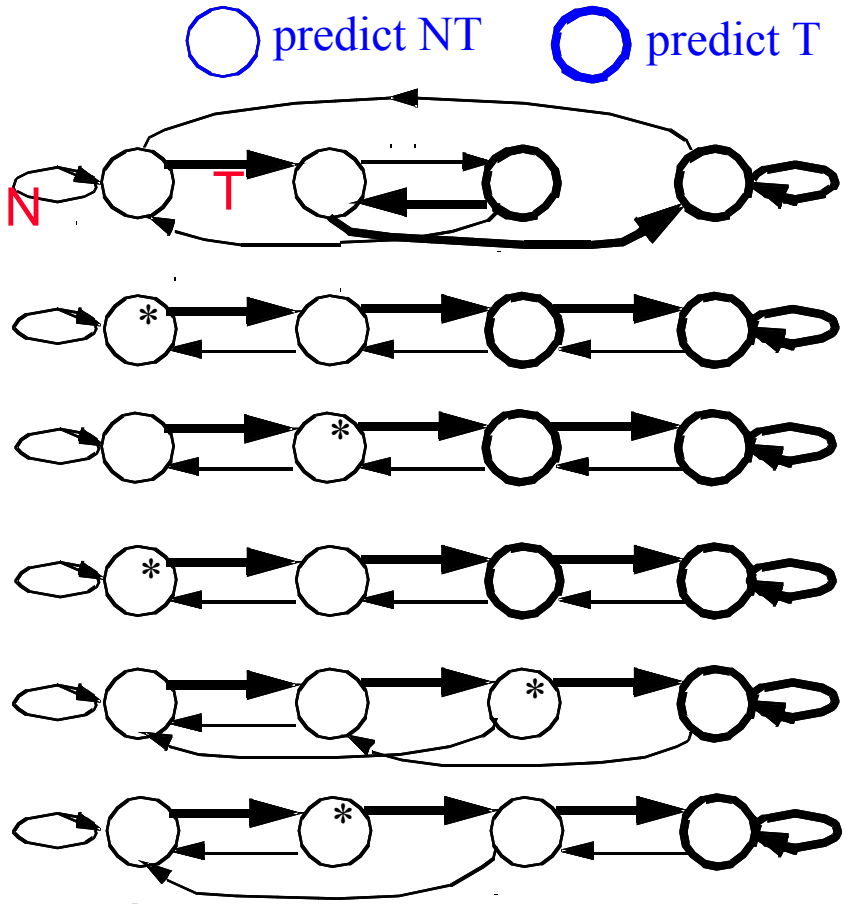


- Combining prediction accuracy with BTB hit rate (86.5% for 128 sets of 4 entries each), branch prediction can provide the net prediction accuracy of approximately 80%. This implies a 5-20% performance enhancement.

Exhaustive Search for Optimal Predictors *[Nair, 1992]*

- There are 2^{20} possible state machines of 2-bit predictors
- Pruning uninteresting and redundant machines leaves 5248
- It is possible to exhaustively search and find the *optimal* predictor for a benchmark

Benchmark	Best Pred. %
spice2g6	97.2
doduc	94.3
gcc	89.1
espresso	89.1
li	87.1
eqntott	87.9



Saturation Counter is near optimal in all cases!

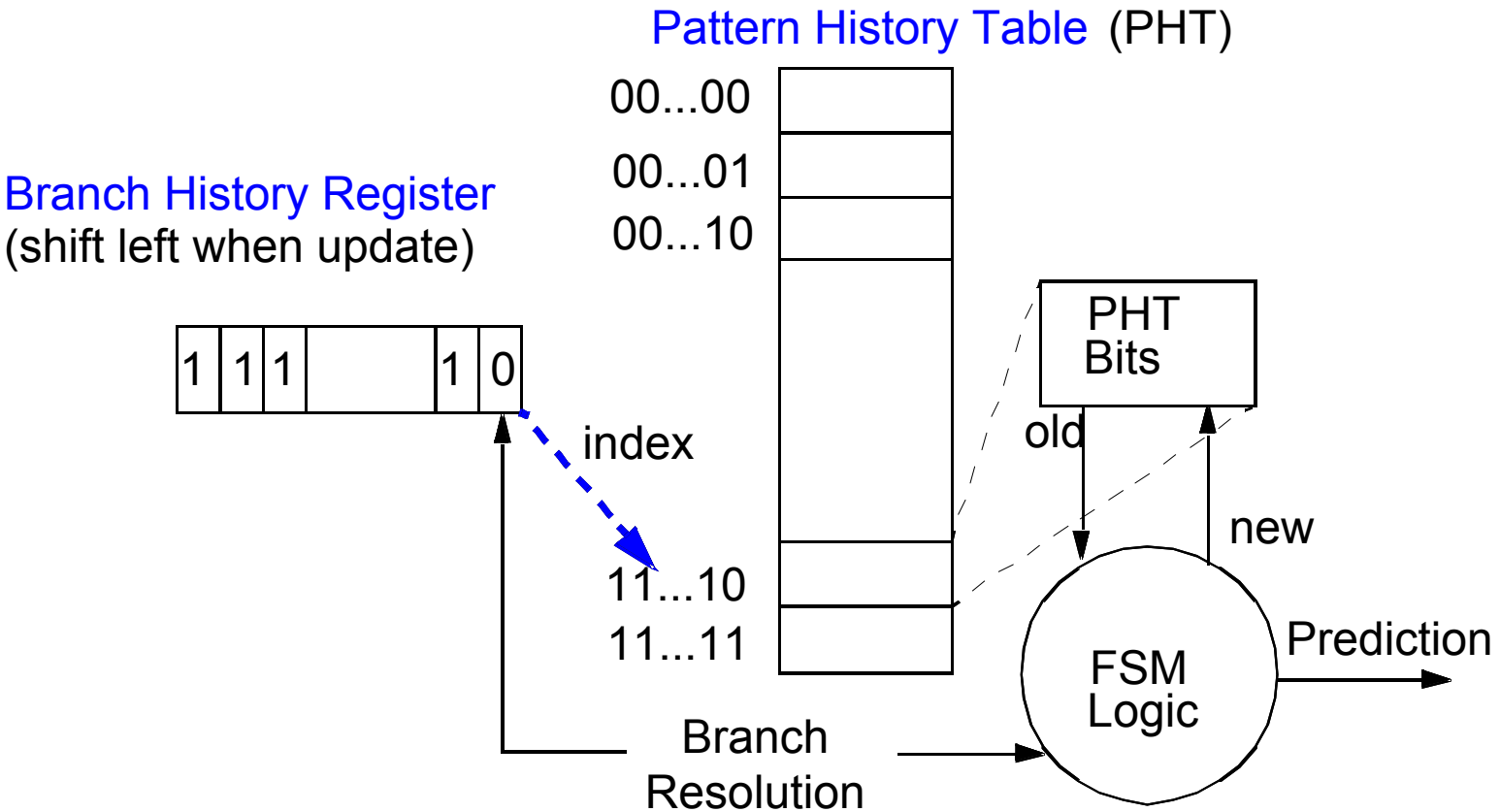
Number of Counter Bits Needed

Benchmark	Prediction Accuracy (Overall CPI Overhead)			
	3-bit	2-bit	1-bit	0-bit
spice2g6	97.0 (0.009)	97.0 (0.009)	96.2 (0.013)	76.6 (0.031)
doduc	94.2 (0.003)	94.3 (0.003)	90.2 (0.004)	69.2 (0.022)
gcc	89.7 (0.025)	89.1 (0.026)	86.0 (0.033)	50.0 (0.128)
espresso	89.5 (0.045)	89.1 (0.047)	87.2 (0.054)	58.5 (0.176)
li	88.3 (0.042)	86.8 (0.048)	82.5 (0.063)	62.4 (0.142)
eqntott	89.3 (0.028)	87.2 (0.033)	82.9 (0.046)	78.4 (0.049)

- Branch history table size: Direct-mapped array of 2k entries
- Programs, like gcc, can have over 7000 conditional branches
- In collisions, multiple branches share the same predictor
- Variation of branch penalty with branch history table size level out at 1024

Global Branch Prediction

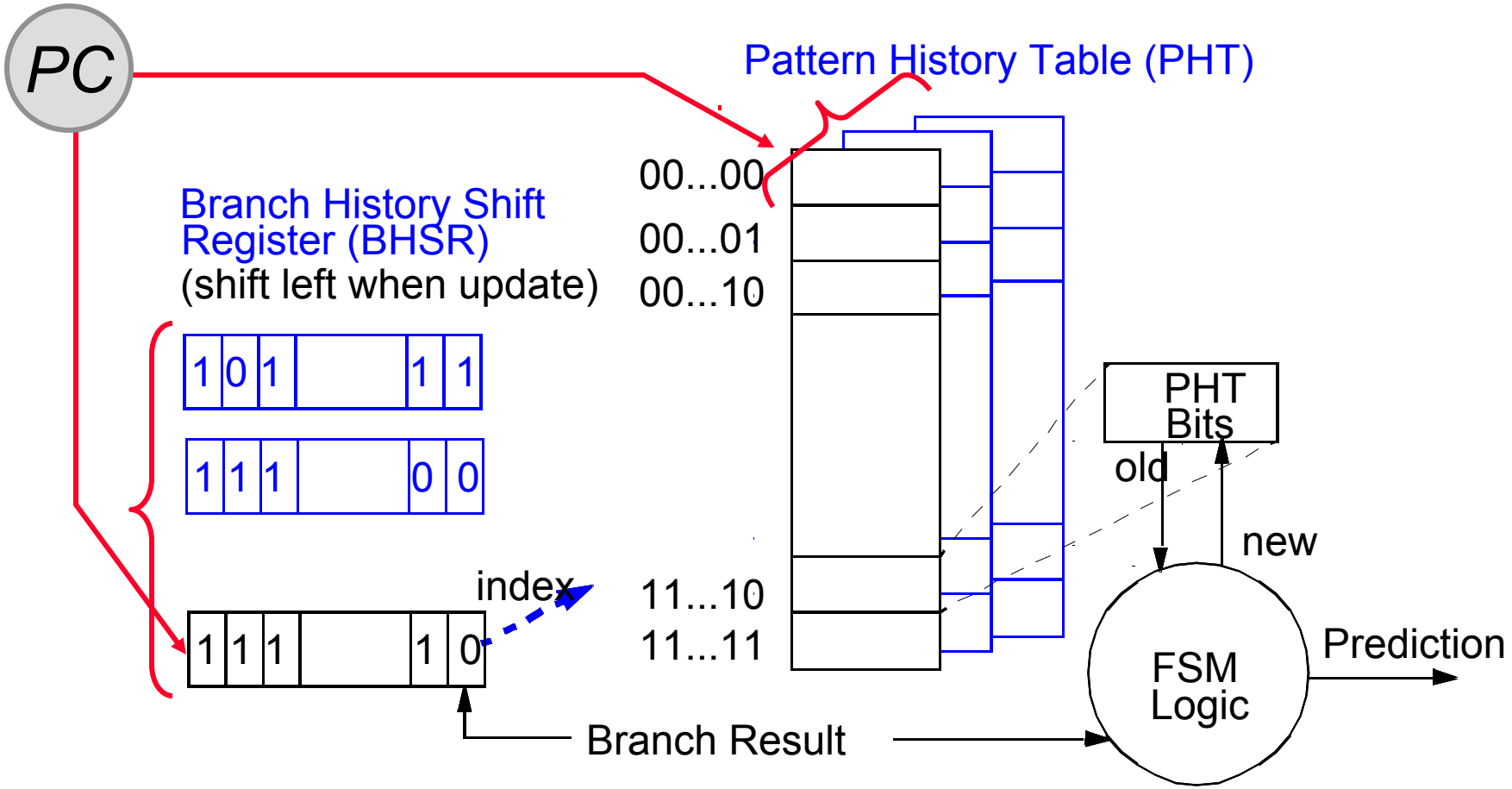
- So far, the prediction of each static branch instruction is based solely on its own past behavior and not the behaviors of other neighboring static branch instructions



2-Level Adaptive Prediction [Yeh & Patt]

- Two-level adaptive branch prediction
  - 1st level: History of last *k* (dynamic) branches encountered
  - 2nd level: branch behavior of the last *s* occurrences of the specific pattern of these *k* branches
  - Use a Branch History Register (BHR) in conjunction with a Pattern History Table (PHT)
- Example: (*k*=8, *s*=6)
  - Last *k* branches with the behavior (11100101)
  - *s*-bit History at the entry (11100101) is [101010]
  - Using history, branch prediction algorithm predicts direction of the branch
- Effectiveness:
  - Average 97% accuracy for SPEC
  - Used in the Intel P6 and AMD K6

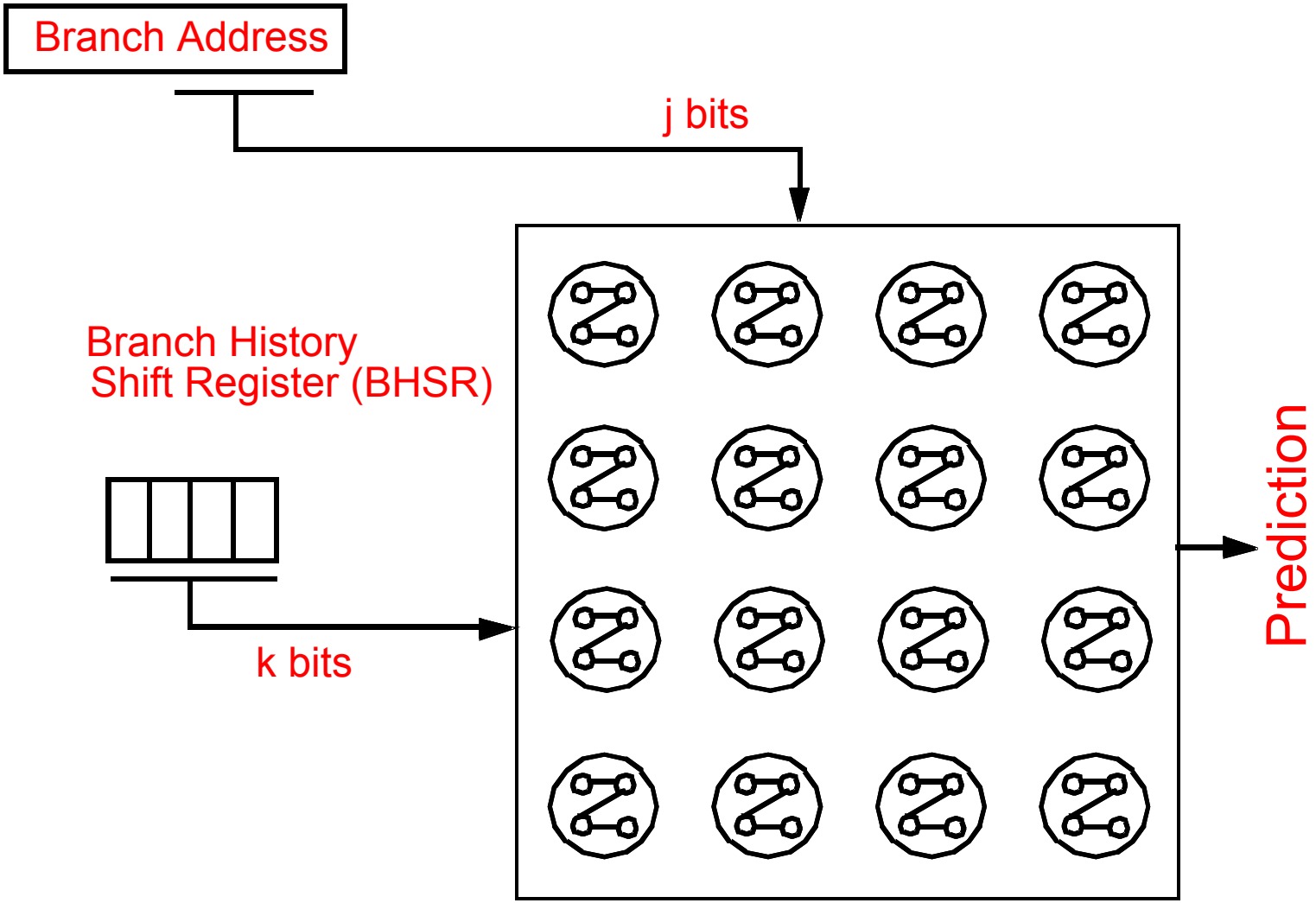
Nomenclature: {G,P}A{g,p,s}



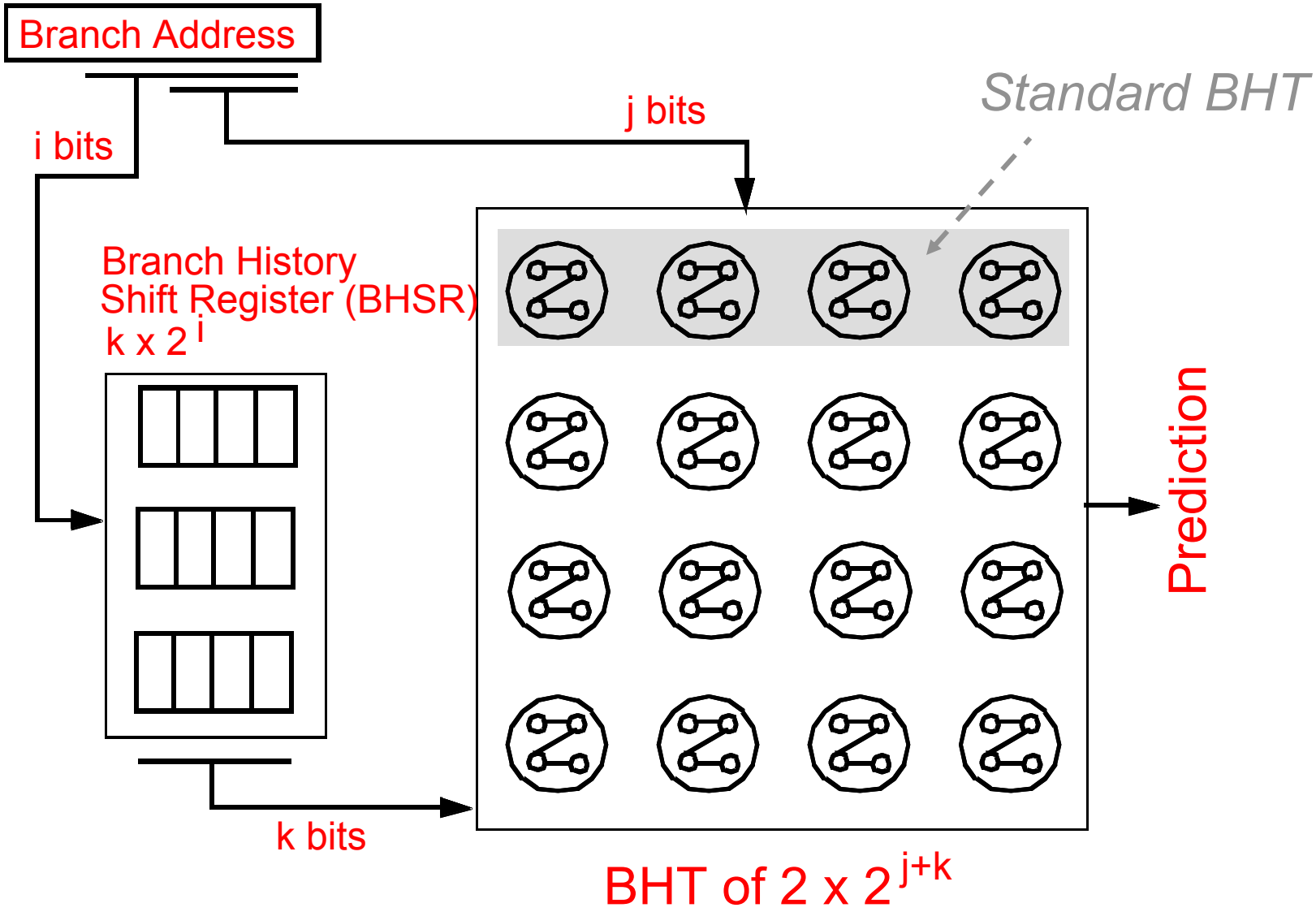
To achieve 97% average prediction accuracy:

G (1) BHR: 18 bits;	g (1) PHT: 2^{18} x 2 bits	*total = 524 kbits*
P (512x4) BHR: 12 bits;	g (1) PHT: 2^{12} x 2 bits	*total = 33 kbits*
P (512x4) BHR: 6 bits;	s (512) PHT: 2^{6} x 2 bits	*total = 78 kbits*

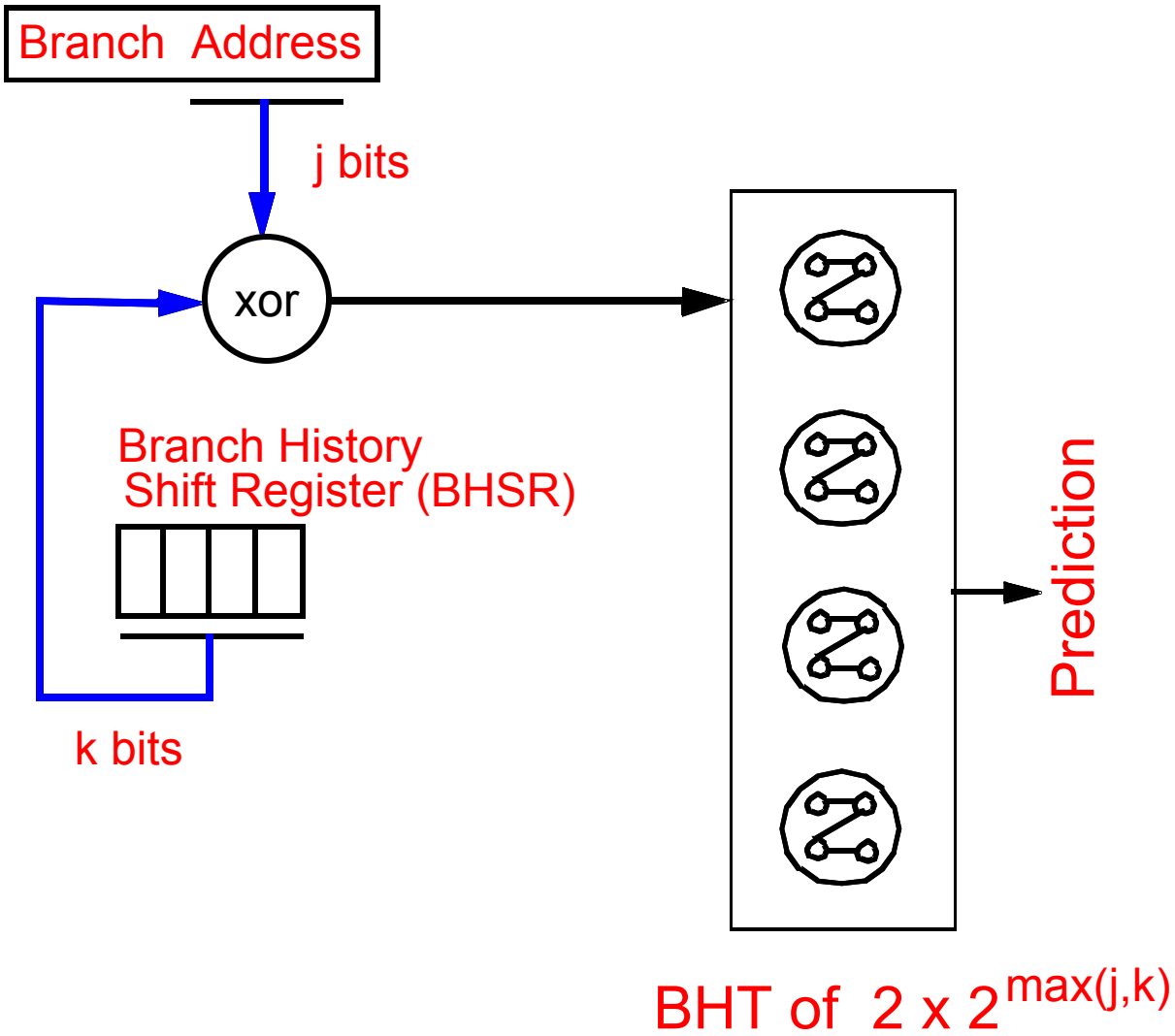
Global BHSR Scheme (GAs)



Per-Branch BHSR Scheme (PAs)



Gshare Branch Prediction [McFarling]



Other Schemes

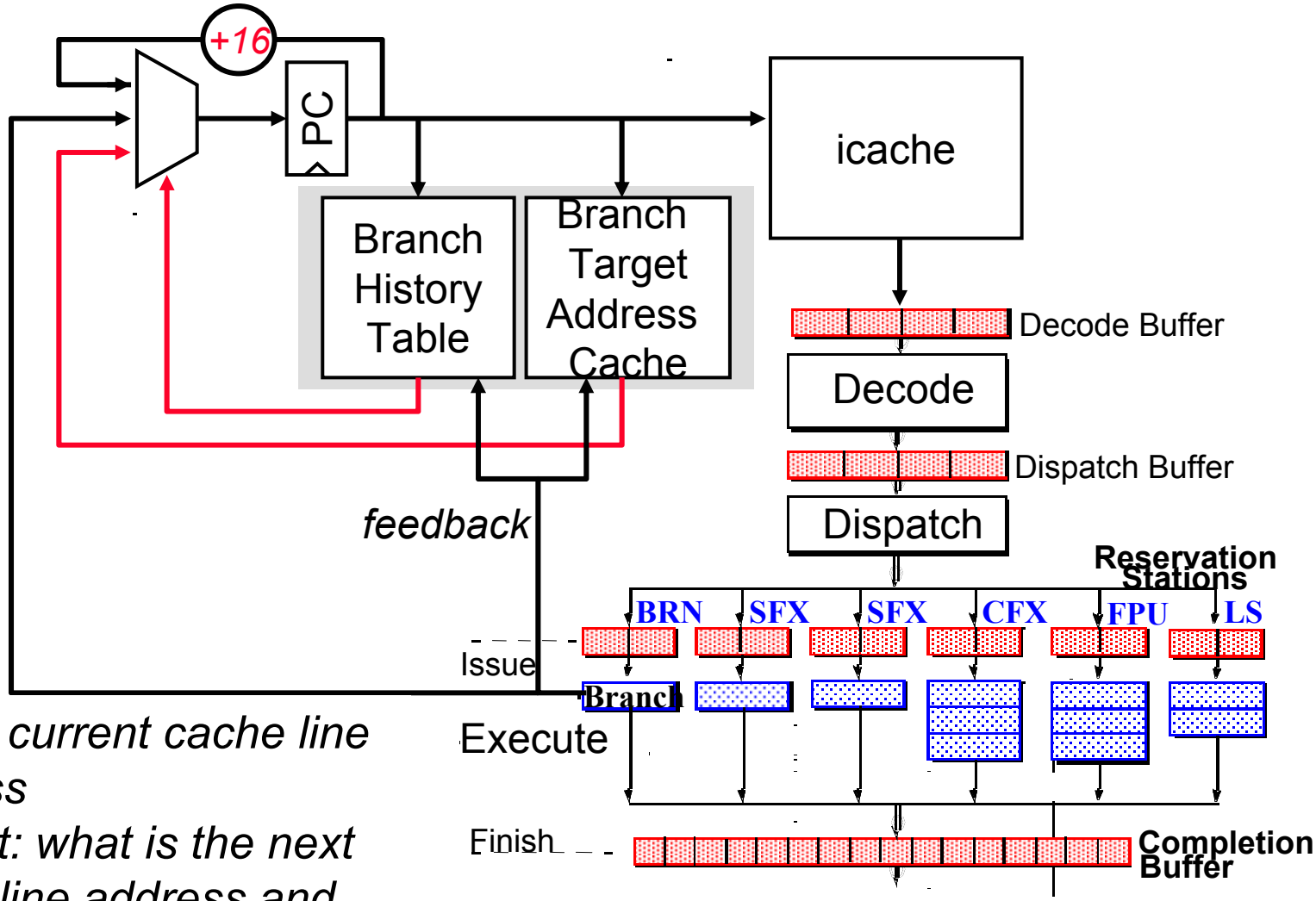
- Function Return Stack
  - Register indirect targets are hard to predict from branch history
  - Register indirect branches are mostly used for function returns

  ⇒ 1. Push the return address onto a stack on each function call

  2. On a reg. indirect branch, pop and return the top address as prediction
- Combining Branch Predictors
  - Each type of branch prediction scheme tries to capture a particular program behavior
  - May want to include multiple prediction schemes in hardware
  - Use another history-based prediction scheme to "predict" which predictor should be used for a particular branch

You get the best of all worlds. This works quite well

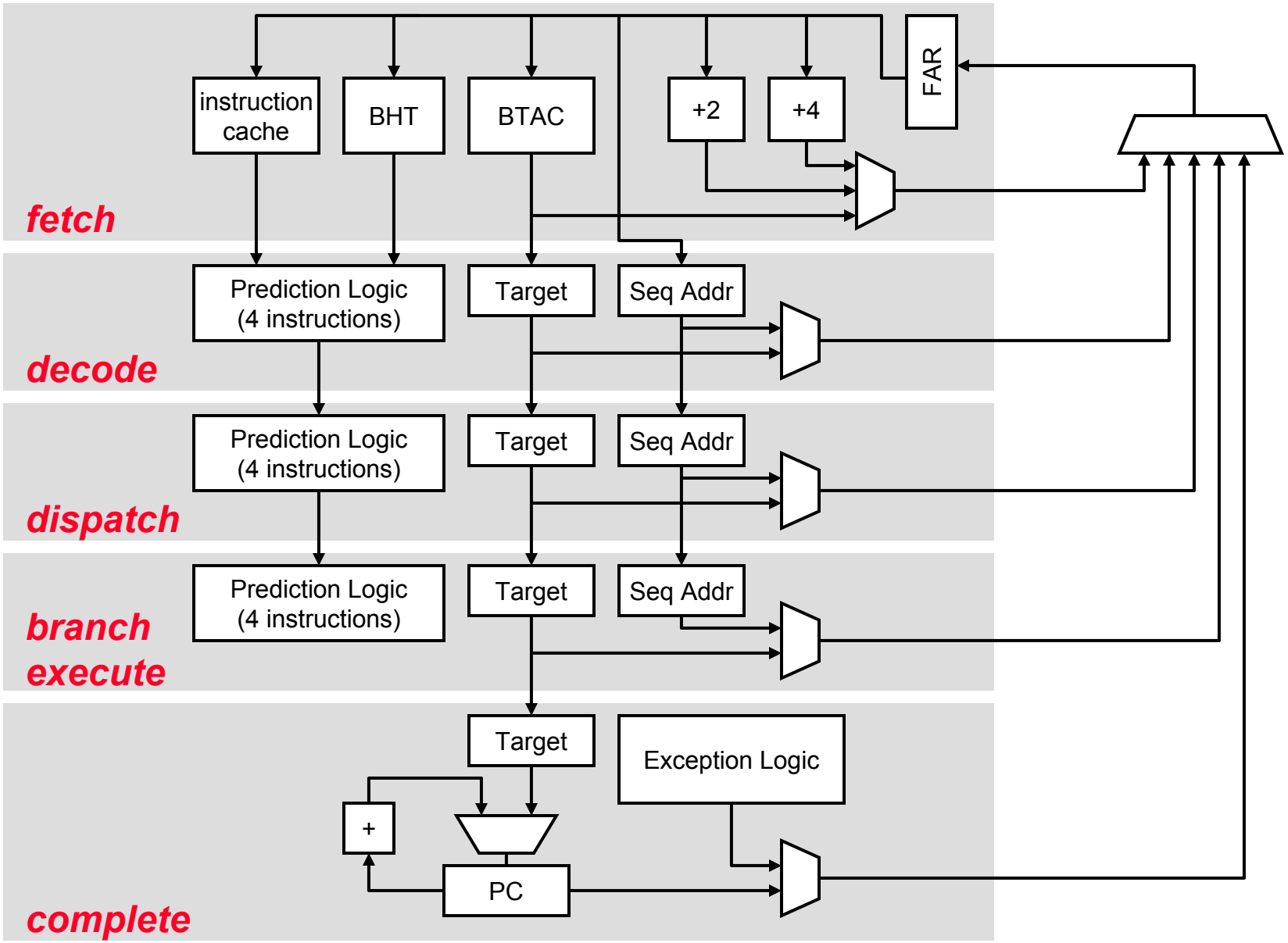
BTB for Superscalar Fetch



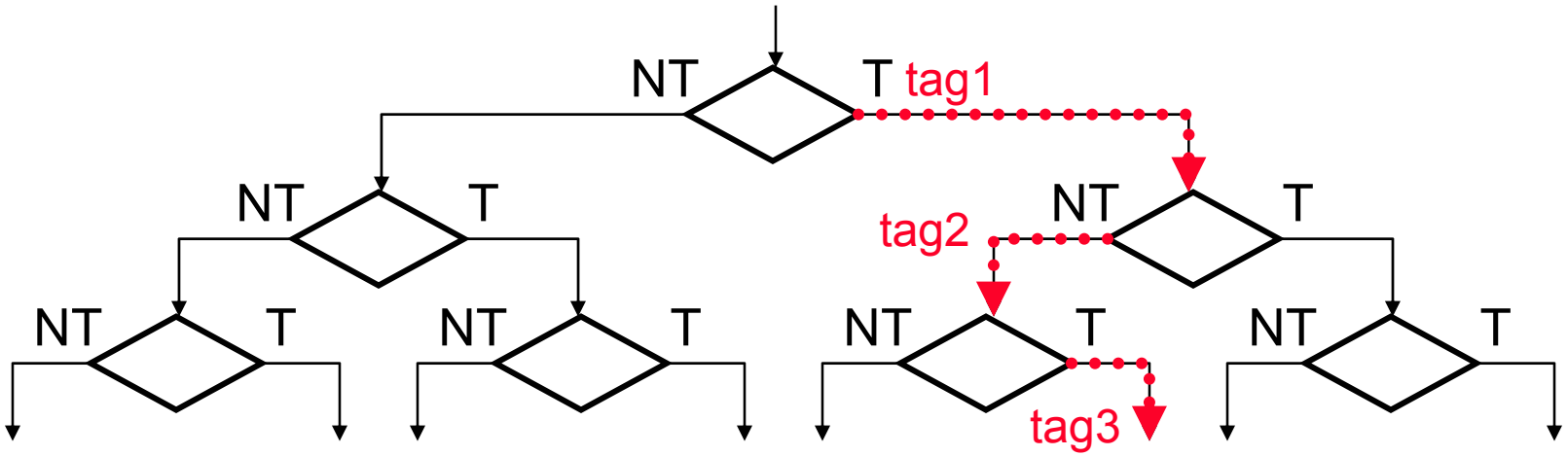
BTB input: current cache line address

BTB output: what is the next cache line address and how many words of the current line is useful

PPC 604 Fetch Address Generation

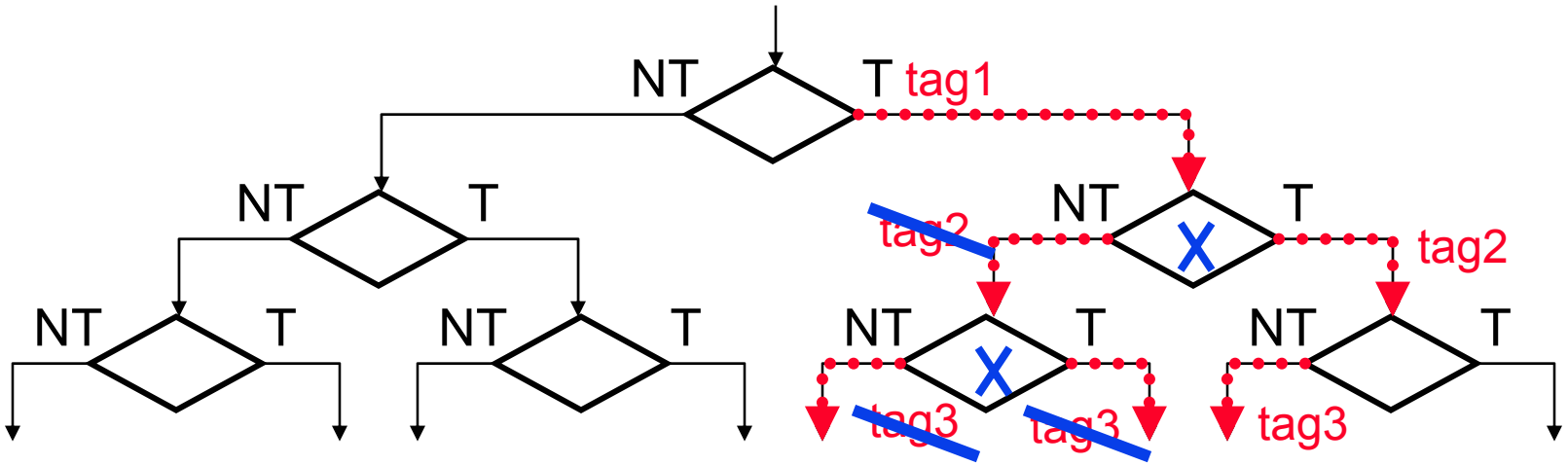


Control Flow Speculation



- Leading Speculation
  - Tag speculative instructions
  - Advance branch and following instructions
  - Buffer addresses of speculated branch instructions

Mis-speculation Recovery



- Eliminate Incorrect Path
  - Must ensure that the mis-speculated instructions produce no side effects
- Start New Correct Path
  - Must have remembered the alternate (non-predicted) path

Mis-speculation Recovery

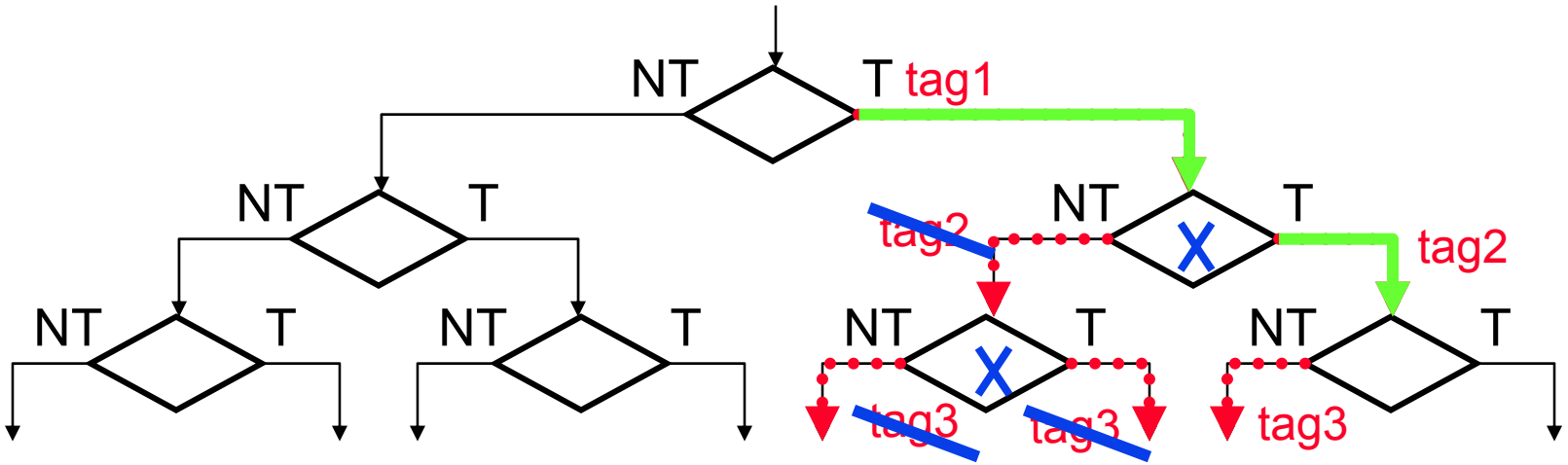
- Eliminate Incorrect Path
  - Use *branch* tag(s) to deallocate completion buffer entries occupied by speculative instructions (now determined to be mis-speculated).
  - Invalidate all instructions in the decode and dispatch buffers, as well as those in reservation stations

How expensive is a misprediction?

- Start New Correct Path
  - Update PC with computed branch target (if it was predicted NT)
  - Update PC with sequential instruction address (if it was predicted T)
  - Can begin speculation once again when encounter a new branch

How soon can you restart?

Trailing Confirmation

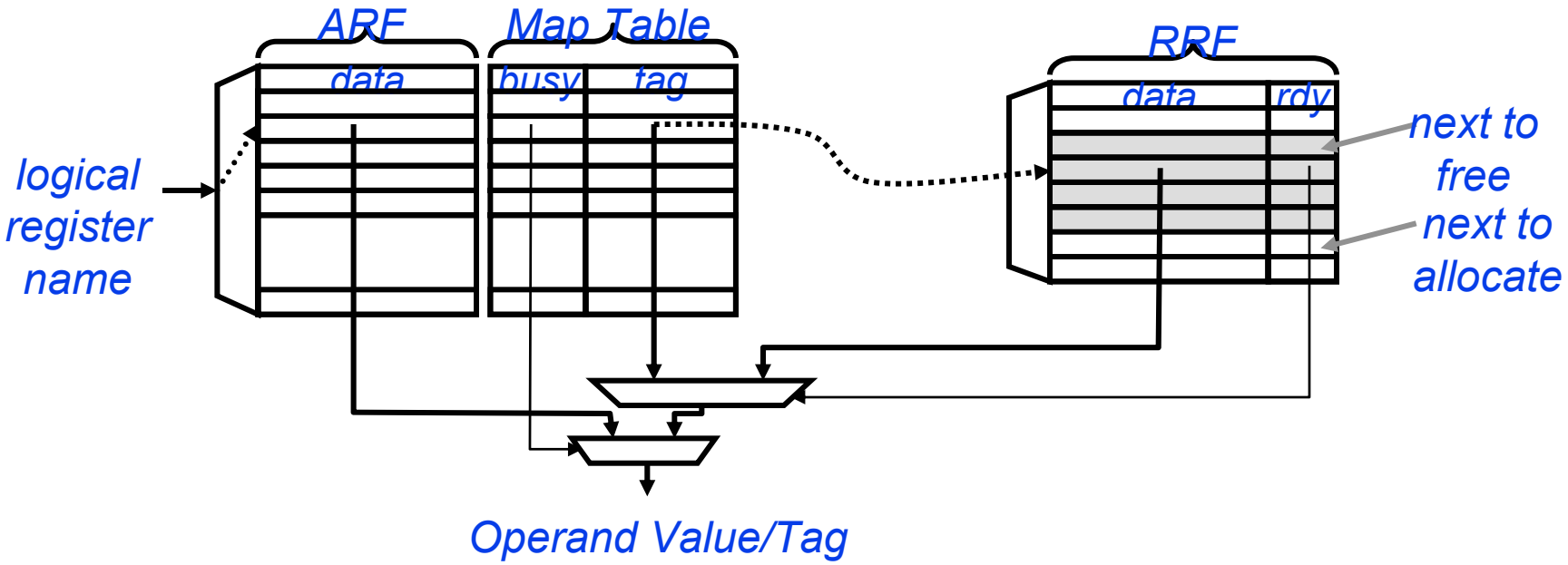


- Trailing Confirmation
  - When branch is resolved, remove/deallocate speculation tag
  - Permit completion of branch and following instructions

Fast Branch Rewind and Restart:
Metaflow DRIS

- Discard all DRIS entries (and corresponding operations) younger than the mispredicted branches
- Can restart immediately from the corrected branch target because the DRIS has sufficient information (rename & value) to continue from where left off
- Works with nested mispredictions!!

Rewinding/Flushing of Rename Table



- To reinitiate renaming:
  - wait for all instructions older than the rewind point to drain clear of the pipeline and then reset register remapping to null

    Long restart latency
  - Reorder buffer has to remember how to restored the map table to the point of the mispredicted branch

    Complicated multi-cycle logic
  - Cache rename map after branch prediction

Impediments to Wide Fetching

- Average Basic Block Size
  - integer code: 4-6 instructions
  - floating-point code: 6-10 instructions
- Branch Prediction Mechanisms
  - must make multiple branch predictions per cycle
  - potentially multiple predicted taken branches
- Conventional I-Cache Organization
  - must fetch from multiple predicted taken targets per cycle
  - must align and collapse multiple fetch groups per cycle

More to come: Trace Caching!!