

# Lecture 14: Dealing with Branches

COS 471a, COS 471b / ELE 375

Computer Architecture and Organization

Princeton University  
Fall 2004

Prof. David August

1

## Program Notes

### Tests Back Today

Review the answer sheet before requesting a regrade!

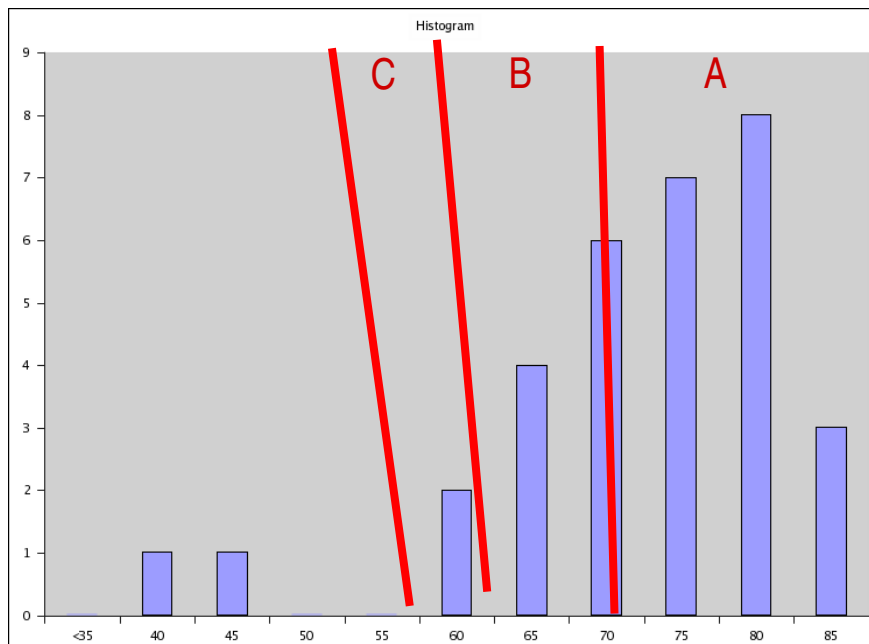
See:

David August - Questions I, II  
Jonathan Chang – Question III  
Neil Vachharajani - Question IV

Check for answer sheet on web site soon.  
Go over test?

2

## Distribution (Approximate Grades)



3

## Distribution

- Mean: 70.1 (A-/B+)
- Median: 73.5 (A)
- Standard Deviation: 10.7

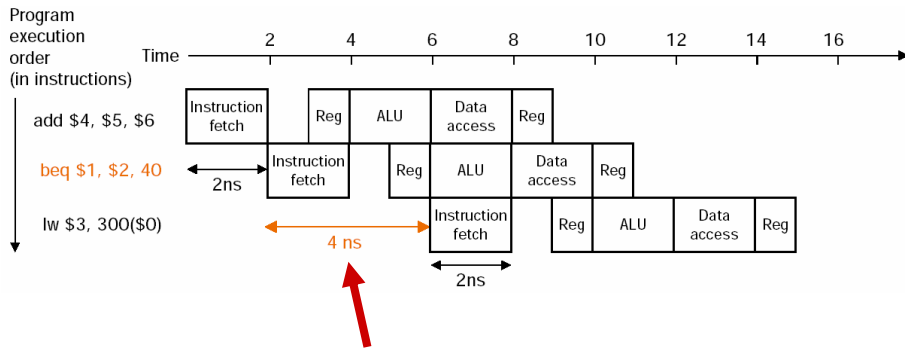
4

## FLASHBACK: Pipeline Hazards

### Control Hazards

- What is the next instruction?
- Branch instructions take time to compute this.

Stall, Predict, or Delay:



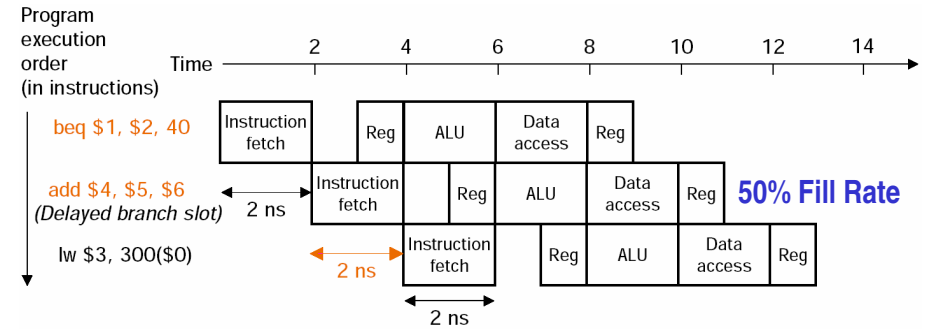
Pipeline Stall - only 1 cycle/stage delay...

## FLASHBACK: Pipeline Hazards

### Control Hazards

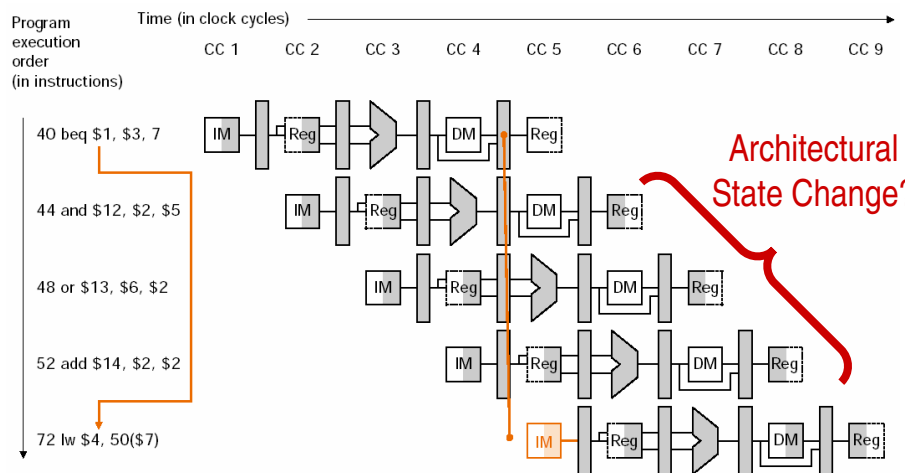
- What is the next instruction?
- Branch instructions take time to compute this.

Delayed Decision (Used in MIPS):



More about Branch Prediction/Delayed Branching Later...

## FLASHBACK: What About Control Hazards? (Predict Not-Taken Machine)



We are OK, as long as we squash. Can we reduce delay?

## Cold Realities

Pipelines are much longer (Pentium 4 has +22 stages!)  
 1 branch delay cycle not possible in these designs

### Stall

- Makes a Pentium 4 run about as fast as an 80486
- What is the point of pipelining if it is all fill/drain?

### Branch Delay Slots

- Very rare to be able to fill more than 1 slot
- Ugly architecture - exposes implementation

Delay slots for a processor that can handle multiple instructions simultaneously?

## Cold Realities

Pipelines are much longer (Pentium 4 has +22 stages!)

Stall - Nope

Branch Delay Slots - Nope

Predict Not Taken

- Roughly 33% of branches are taken

Is a 67% prediction rate good enough?

11

## The Importance of High Prediction Rates

- In real non-scientific codes, 1 in every 4 or 5 instructions is a branch.
- Misprediction penalties are high
  - 17 wasted cycles in Pentium 4
  - 7 wasted cycles in Alpha 21264

Example:

- Pipelining throughput (ideal) = 1 CPI
- 67% prediction rate, 1 in 4 instructions a branch
- 20 cycle penalty

$$0.25 \text{ br/i} * 0.33 \text{ miss/br} * 20 \text{ c/miss} + 1 \text{ c/i} = 2.67 \text{ c/i}$$
$$\text{Slowdown} = 2.67x$$

12

## The Importance of High Prediction Rates

- In real non-scientific codes, 1 in every 4 or 5 instructions is a branch.
- Misprediction penalties are high
- Modern machines execute multiple instructions per cycle

Wide-Issue Example:

- Pipelining throughput (ideal) = 4 IPC (0.25 CPI)
- 67% prediction rate, 1 in 4 instructions a branch
- 20 cycle penalty

$$0.25 \text{ br/i} * 0.33 \text{ miss/br} * 20 \text{ c/miss} + 0.25 \text{ c/i} = 1.9 \text{ c/i}$$
$$\text{Slowdown} = 7.6x$$

13

## How do we get better prediction rates?

### Better Prediction

- Predictions can be made by the **Hardware** or by the **Compiler**
- Predictions can set before execution (**Static**) or adapt during execution (**Dynamic**)

	Hardware	Compiler
Static	Predict Not Taken	
Dynamic		

Let's Look Here Next For No Apparent Reason

14

## Compiler-Static Prediction

Key idea: Compiler indicates a prediction to the HW

### Communication Schemes

- Use a **prediction bit** in your branch encoding
  - Bit = 1 à predict taken
  - Bit = 0 à predict not-taken
- Backward taken, forward not taken (BTFNT)
  - Negative displacement à predict taken
  - Positive displacement à predict not-taken

Why BTFNT and not FTBNT?  
Does it really matter?

15

## Compiler-Static Prediction

Key idea: Compiler indicates a prediction to the HW

### Compiler Prediction Methods

- Profiling
  - Run program and observe
- Rule-based
  - Loops do
  - Exits don't

Issues?

Why?

16

## How do we get better prediction rates?

### Better Prediction

- Predictions can be made by the **Hardware** or by the **Compiler**
- Predictions can set before execution (**Static**) or adapt during execution (**Dynamic**)

	Hardware	Compiler
Static	Predict Not Taken	Prediction Bit, BTFNT
Dynamic		

Advantages of these quadrants?  
Disadvantages?

17

## Dynamic Branch Prediction

Key idea: Branches have personality

Question: What do you think the branch will do next?  
(1 = taken, 0 = not taken)

Branch History: 111 ?

Branch History: 101010101 ?

Branch History: 110011001100110011 ?

Branch History: 0000000001111111111 ?

Branch History: 11111101111 ?

How Do We Capture This?

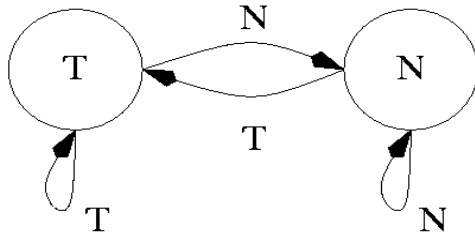
18

## Hardware-Dynamic Branch Prediction

### Automata

Consider history of 111111 or 00000

**Automaton: Last-Time (LT)**



When is a Last-Time predictor better than a compiler static predictor?  
What happens the first time the branch is seen?

19

## Hardware - Dynamic Branch Prediction

### Automata - More States à Better

#### Other Automata

Consider another pattern with Last-Time:

- 1110111011101
- Last-Time = 6 misses

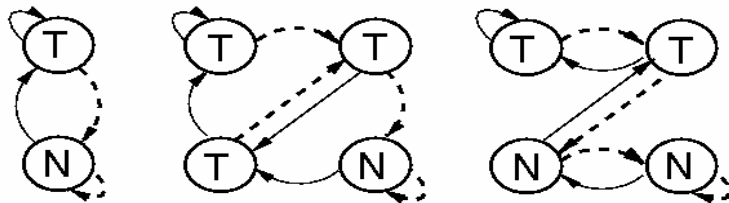
Only 3 misses with 4 states:

State	Meaning	Prediction
00	Strongly Not Taken	Not Taken
01	Weakly Not Taken	Not Taken
10	Weakly Taken	Taken
11	Strongly Taken	Taken

- Used by Alpha 21164, Sun UltraSPARC, MIPS R10000, and others.
- Jim Smith, 1991

20

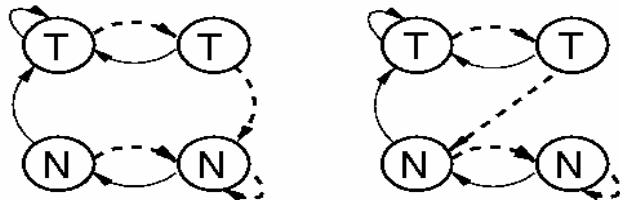
### Automata



Automaton Last-Time (LT)

Automaton A1

Automaton A2  
(2-bit Saturating Up-down Counter)



Automaton A3

Automaton A4

21

### Branch Correlation

- All these automata capture only *Self-Correlation* of branches.

- Branches can be correlated: (From *eqntott*, SPEC92)

```

if (aa == 2)           /* branch 1 */
    aa = 0;
if (bb == 2)           /* branch 2 */
    bb = 0;
if (aa != bb) {       /* branch 3 */
    .....
}
    
```

- If the conditions of branch 1 and branch 2 are TRUE, the condition of branch 3 is FALSE.

Can other-branch history be used to improve prediction?

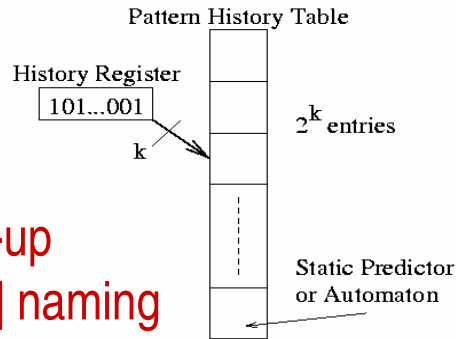
22

# Branch Prediction with History Tables

## Types of Pattern History Tables

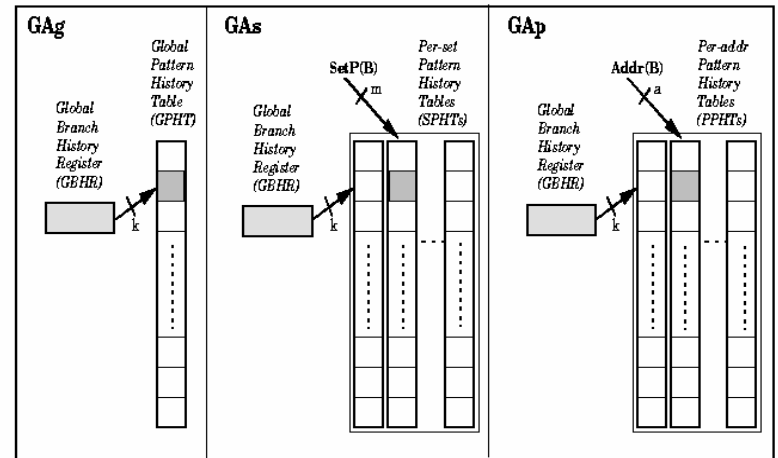
Pattern History Tables can be:

- *Global* - one predictor for each pattern shared with all branches.
- *Per-address* - one predictor for each pattern for each branch.
- *per-Set* - one predictor for each pattern for each set of branches.

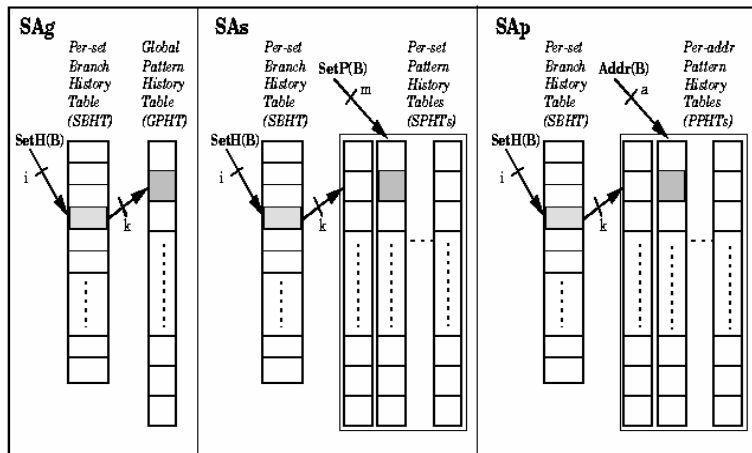


Warm-up  
[GPS]A[gps] naming

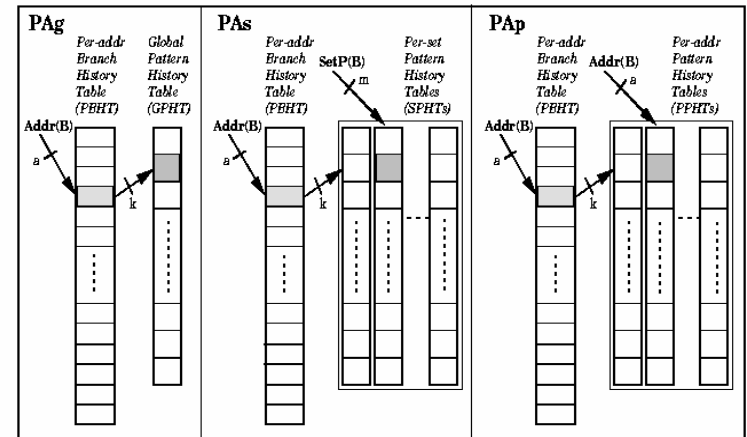
## The Global History Adaptive Schemes



## The Per-Set History Adaptive Schemes



## The Per-Address History Adaptive Schemes



### Other Two-Level Schemes Exist

#### gshare

- A single global branch history.
- A single branch history table.
- Index into branch history table is computed as the XOR of the branch address and the global branch history.

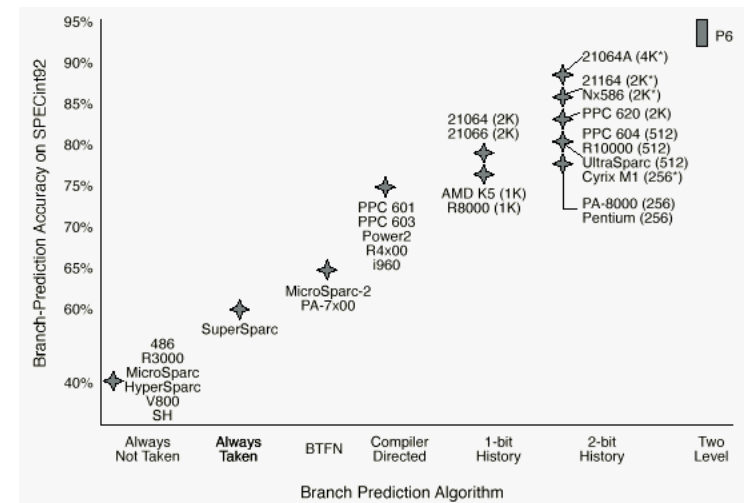
#### gselect

- Similar to gshare.
- Index into branch history table is computed as the CONCATENATION of the branch address and the global branch history.

#### others

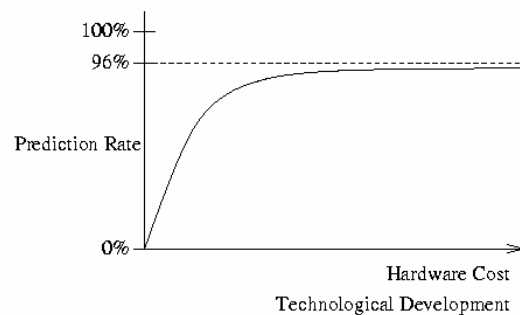
- yags - yet another gshare
- variable history length predictors

## Branch Prediction Comparisons



Source: Microprocessor Report

## Limits of Hardware Dynamic Branch Prediction Information Theoretic Viewpoint



- Consider branch prediction history.
- This branch history contains truly independent events.
- You can't predict the next white noise sample using knowledge gained from previous samples.
- Branch prediction using branch history has a theoretic limit.
- Trends indicate most prediction information extracted.

HPCA Conference in 1994

Compiler-Synthesized Branch Prediction

### Conventional Branch Prediction Techniques

	Hardware	Compiler
Static	Uninteresting	Prediction bit, BTFNT
Dynamic	2-bit Counter, 2-Level	Unexplored, Focus of this Work

- **Compiler/Static branch prediction**
  - Compiler controlled - compile time decision
  - (+) No hardware limits - every branch has a real prediction
  - (+) Low cost
  - (-) Prediction cannot vary during program execution
  - (-) Performs poorly for unbiased branches
- **Hardware/Dynamic branch prediction**
  - Hardware controlled - state machine makes prediction
  - (+) Accuracy - varies during run-time
  - (-) Hardware has diminishing returns
  - (-) Area, power

## Compiler Dynamic

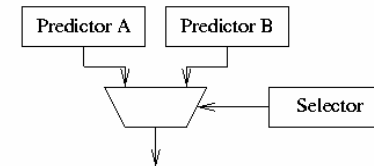
You tell me...

How does this overcome limits of hardware dynamic?

31

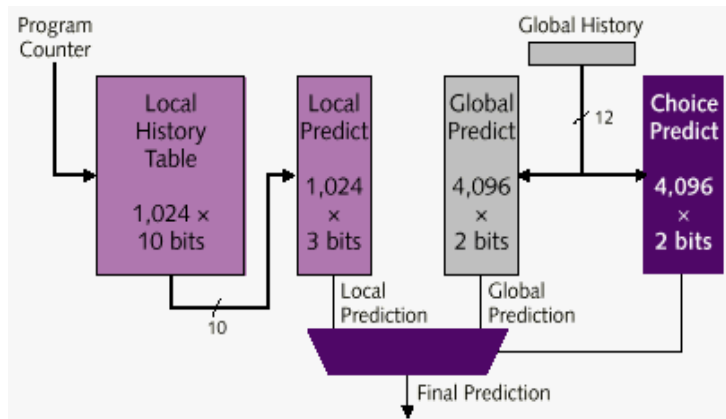
## Hybrid/Tournament Predictors

- If one is good, why not two, three, etc.?
- Hybrid (or multiple-scheme) branch prediction was proposed by Scott McFarling, WRL 1993
- Different predictors are adept at capturing different branch correlation. (global vs. local history)
- Use compiler or hardware to select predictor at run-time.
- Predictor selection mechanisms are very similar to branch predictors - predict which predictor is going to predict the branch correctly.



33

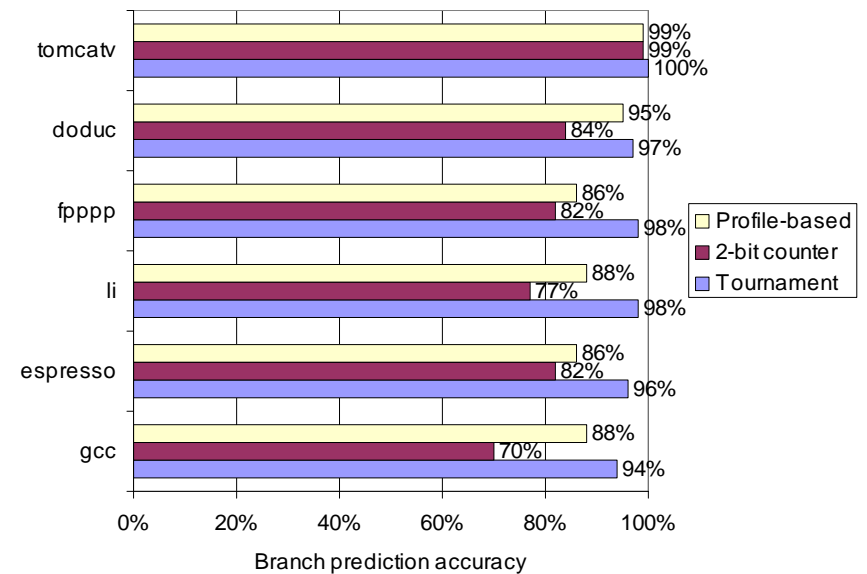
## 21264 Branch Prediction Logic



- 35Kb of prediction information
- 2% of total die size

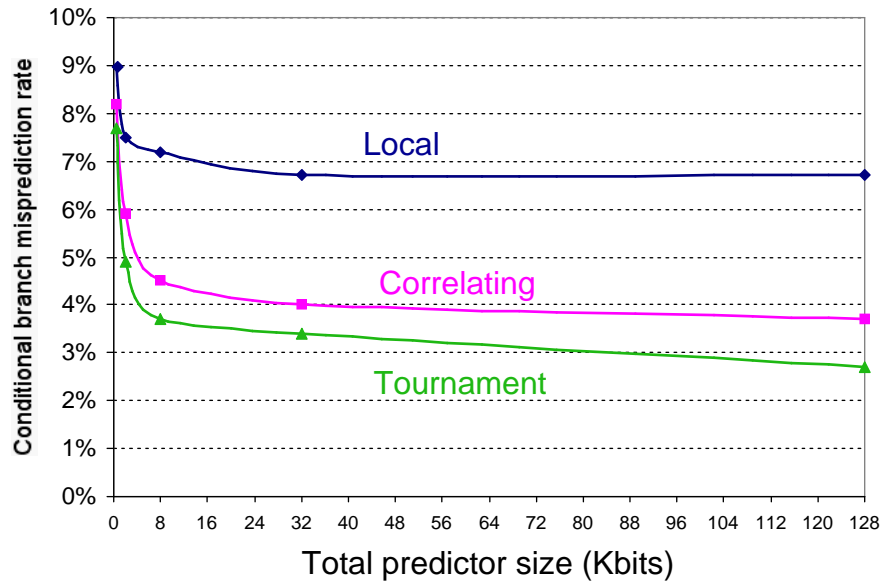
34

## Accuracy of Branch Prediction



35

## Accuracy v. Size (SPEC89)



36

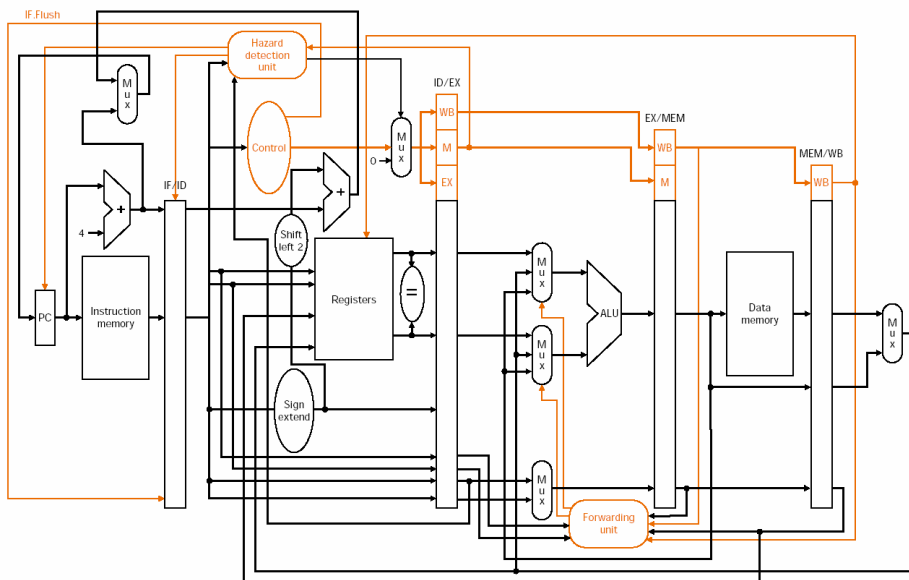
## FLASHBACK: Reduce Branch Delay

1. Move branch address calculation to decode stage (from MEM stage)
2. Move branch decision up (Harder)
  - Bitwise-XOR, test for zero
  - Only need Equality testing
  - Much faster: No carry

Everything is done in decode stage!!

38

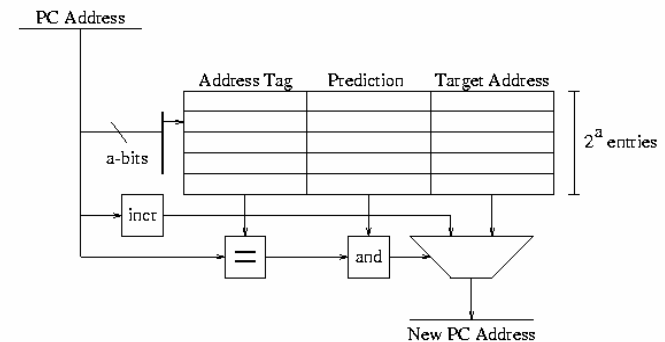
## FLASHBACK: Reduce Branch Delay



39

## The Branch Target Buffer

- The Branch Target Buffer (BTB) is used by the fetch unit to determine the next Program Counter (PC) value.
- Some BTBs are tag-less to reduce table size and speed prediction delivery.



First time a branch is encountered?

40

## Another way to deal with branches

### Eliminate them!!!

- Predication (Intel IA-64/Itanium Processor and others)
- Predication vs. Prediction (Ugh! Try writing a thesis with these words...)
  
- Blackboard...

## Summary

- Branches have personality
- Compiler/HW, Static/Dynamic
- Key ideas:
  - Correlation
  - Prediction accuracy
  - Hardware cost
  - Warm-up
- Hybrid Predictors
- Branch Target Buffers
- Predication