

---

# EEC 581 Computer Architecture

## Lec 5 – Instruction Level Parallelism

**Chansu Yu**  
Electrical and Computer Engineering  
Cleveland State University

### Acknowledgement ...

---

- **Part of class notes are from**
  - David Patterson
  - Electrical Engineering and Computer Sciences
  - University of California, Berkeley
  
  - <http://www.eecs.berkeley.edu/~pattsrn>
  - <http://www-inst.eecs.berkeley.edu/~cs252>

## Outline

---

- ILP
- Compiler techniques to increase ILP
- Loop Unrolling
- Static Branch Prediction
- Dynamic Branch Prediction
- Overcoming Data Hazards with Dynamic Scheduling
- Tomasulo Algorithm
- Conclusion

10/4/2007

3

## Overview of Chap. 2 & 3 (again)

---

- Pipelined architecture allows multiple instructions run in parallel (ILP)
- But, it has data and control hazard problems
  
- How can we avoid or alleviate the hazard problems in pipelined architecture?

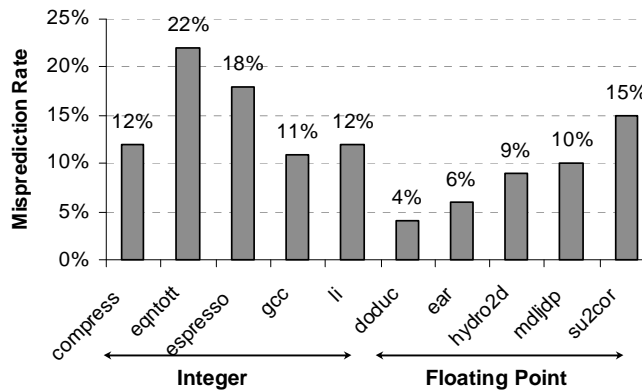
10/4/2007

4

## Static Branch Prediction

- To reorder code around branches, need to predict branch statically when compile
- Simplest scheme is to predict a branch as taken
  - Average misprediction = untaken branch frequency = 34% SPEC

- More accurate scheme predicts branches using profile information collected from earlier runs, and modify prediction based on last run:



10/4/2007

## Dynamic Branch Prediction

- Why does prediction work?
  - Underlying algorithm has regularities
  - Data that is being operated on has regularities
  - Instruction sequence has redundancies that are artifacts of way that humans/compiler think about problems
- Is dynamic branch prediction better than static branch prediction?
  - Seems to be
  - There are a small number of important branches in programs which have dynamic behavior

10/4/2007

6

## Dynamic Branch Prediction

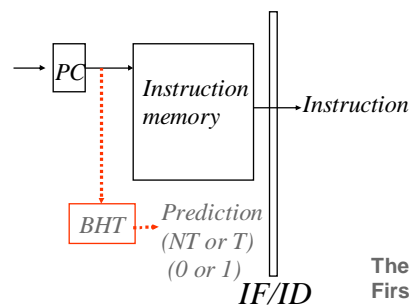
- Performance =  $f(\text{accuracy, cost of misprediction})$
- Branch History Table: Lower bits of PC address index table of 1-bit values
  - Says whether or not branch taken last time
  - No address check
- Problem: in a loop, 1-bit BHT will cause two mispredictions (avg is 9 iterations before exit):
  - End of loop case, when it exits instead of looping as before
  - First time through loop on *next* time through code, when it predicts exit instead of looping

10/4/2007

7

## Dynamic Branch Prediction

- Branch history table



```

addi $1, $0, 10
Loop: ...
...
addi $1, $1, -1
beq $1, $0, Loop
    
```

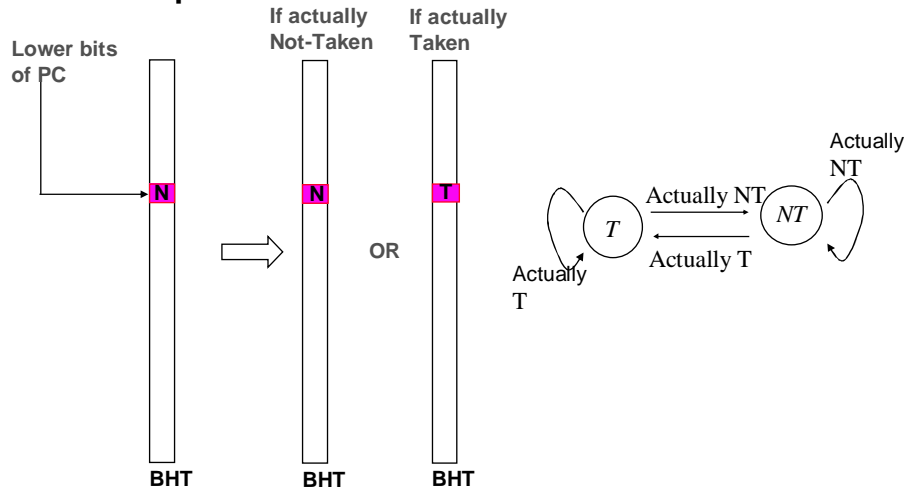
There are 10 loops.  
 First branch: predict ???, Actually Taken  
 Second: predict Taken, Actually Taken  
 Third: Predict Taken, Actually Taken  
 ...  
 Tenth: Predict Taken, Actually Not-Taken  
 ...  
 First: Predict Not-Taken, Actually Taken  
 Prediction accuracy = 80%

10/4/2007

8

# Dynamic Branch Prediction

- 1-bit predictor

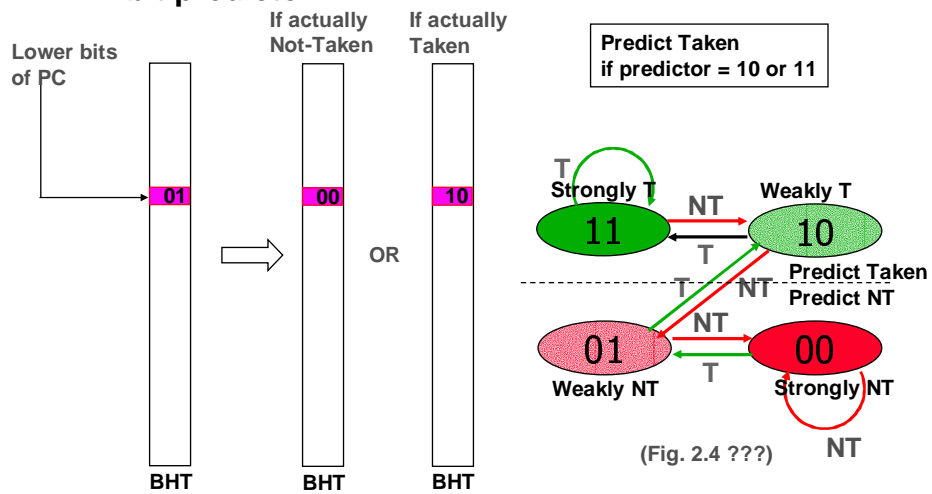


10/4/2007

9

# Dynamic Branch Prediction

- 2-bit predictor

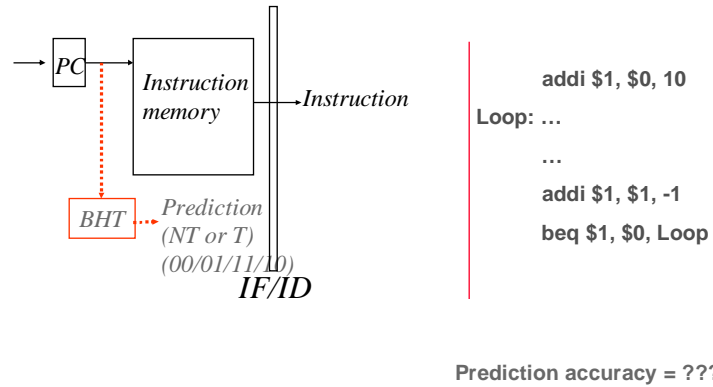


10/4/2007

10

## Dynamic Branch Prediction

- 2-bit predictor

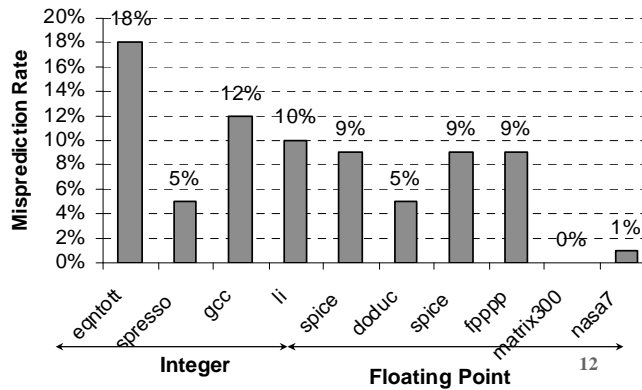


10/4/2007

11

## BHT Accuracy

- Mispredict because either:
  - Wrong guess for that branch
  - Got branch history of wrong branch when index the table
- 4096 entry table:



10/4/2007

## Correlating Branches

---

Code example showing the potential

```
If (d==0)
    d=1;
If (d==1)
    ...
```

Assemble code

```
BNEZ R1, L1
DADDIU R1,R0,#1
L1: DADDIU R3,R1,#-1
    BNEZ R3, L2
L2:
    ...
```

Observation: if (d==0)-branch is taken, then (d==1)-branch is taken too

Idea: taken/not taken of recently executed branches is related to behavior of next branch

10/4/2007

13

## Correlated Branch Prediction

---

- **Idea: Branches are correlated, sometimes**

- Utilize “ $m$ ” most recently executed branches to help predictions
- Maintain  $2^m$  history tables and use a proper  $n$ -bit branch history table based on “ $m$ ” most recent branches

- **( $m,n$ ) predictor**

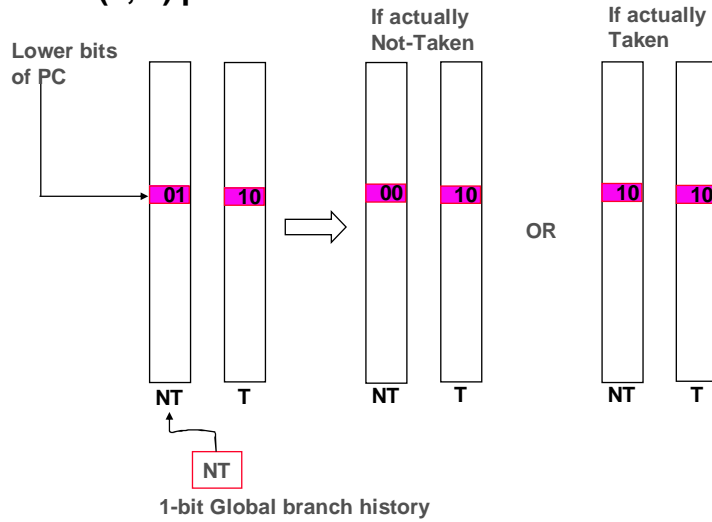
- (1, 2) predictor: 2 history tables, each of which is a 2-bit predictor
- (2, 2) predictor: 4 history tables, each of which is a 2-bit predictor
- (12, 2) predictor: 4K history tables, each of which is a 2-bit predictor

10/4/2007

14

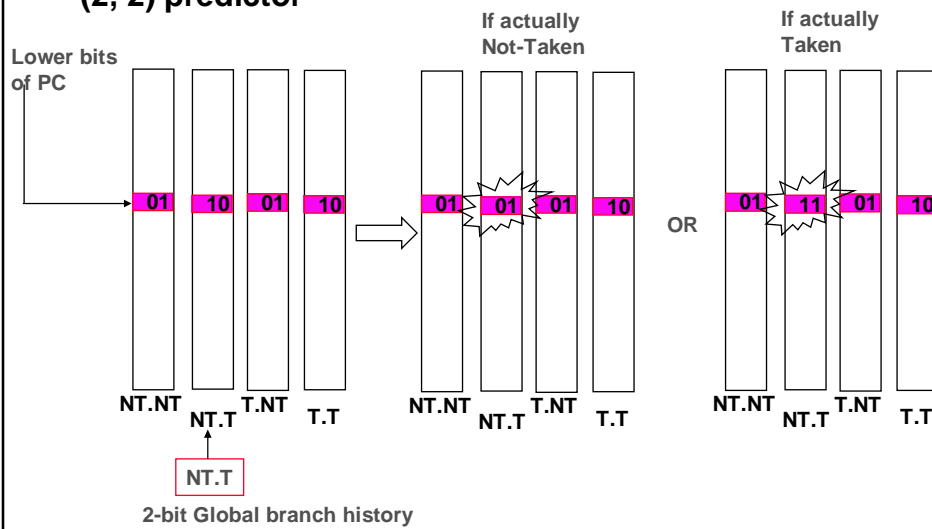
## Correlating Branches

- (1, 2) predictor



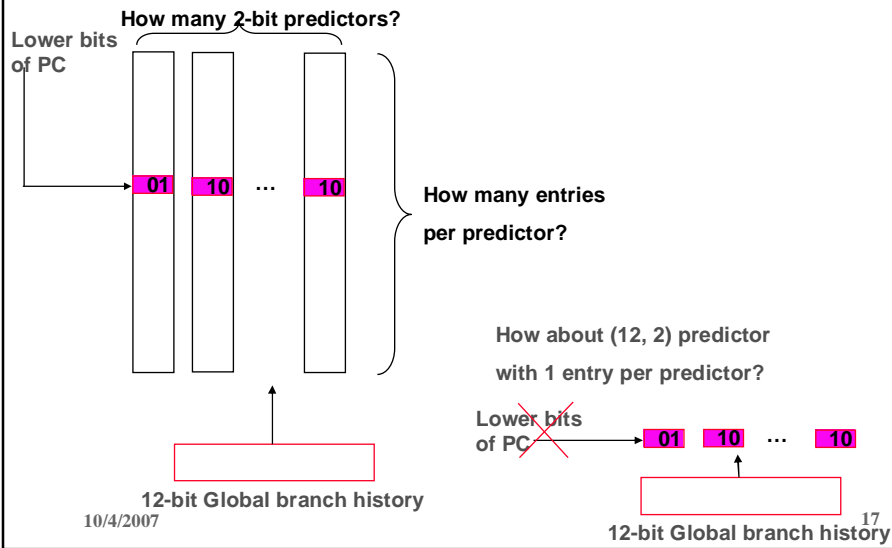
## Correlating Branches

- (2, 2) predictor

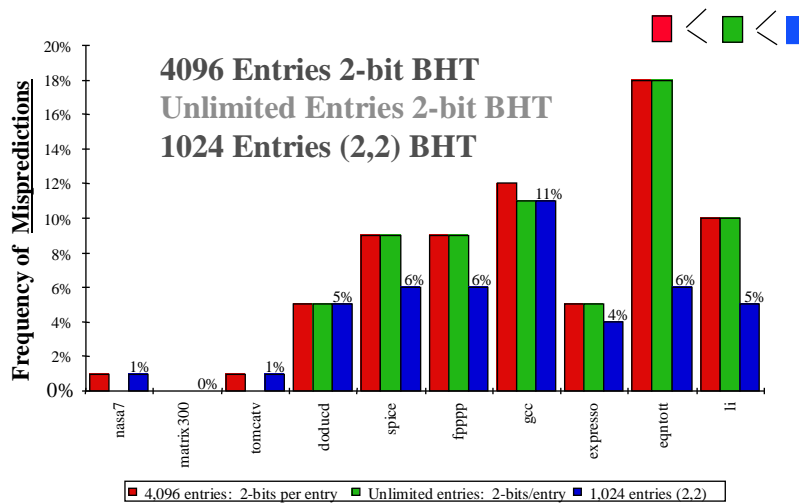


## Correlating Branches

- (12, 2) predictor ???



## Accuracy of Different Schemes



10/4/2007

18

## Tournament Predictors

---

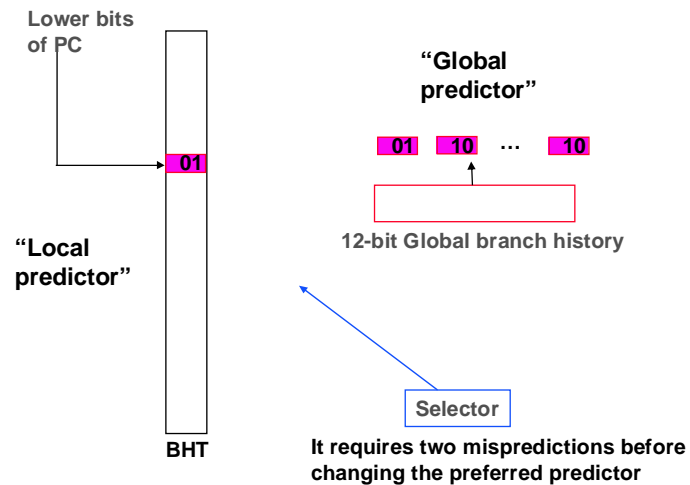
- Multilevel branch predictor
- Use  $n$ -bit saturating counter to choose between predictors
- Usual choice between global and local predictors

10/4/2007

19

## Tournament Predictors

---



10/4/2007

20

## Tournament Predictors

Tournament predictor using, say, 4K 2-bit counters indexed by local branch address. Chooses between:

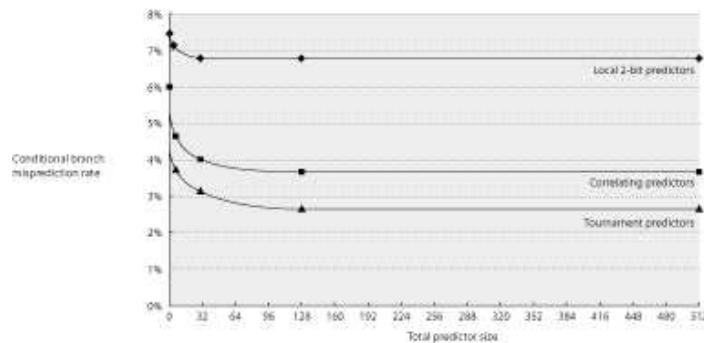
- **Global predictor**
  - 4K entries index by history of last 12 branches ( $2^{12} = 4K$ )
  - Each entry is a standard 2-bit predictor
- **Local predictor**
  - Local history table: 1024 10-bit entries recording last 10 branches, index by branch address
  - The pattern of the last 10 occurrences of that particular branch used to index table of 1K entries with 3-bit saturating counters

10/4/2007

21

## Comparing Predictors (Fig. 2.8)

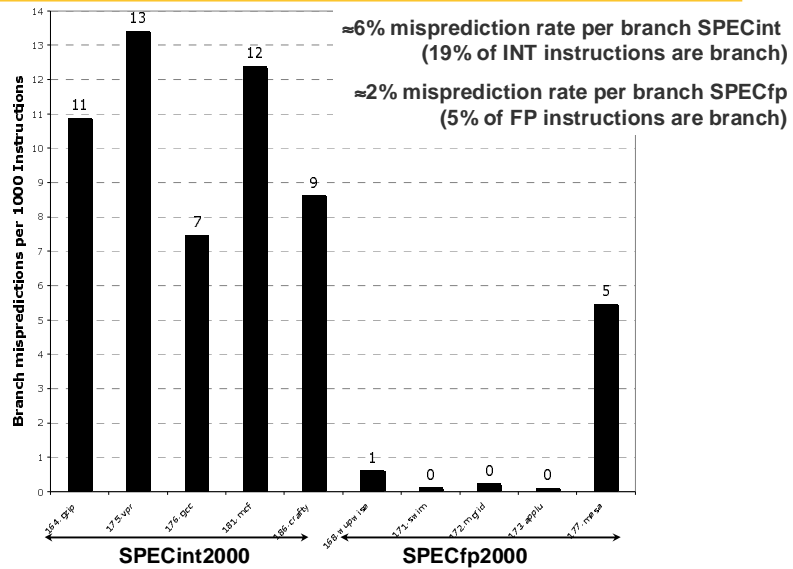
- **Advantage of tournament predictor is ability to select the right predictor for a particular branch**
  - Particularly crucial for integer benchmarks.
  - A typical tournament predictor will select the global predictor almost 40% of the time for the SPEC integer benchmarks and less than 15% of the time for the SPEC FP benchmarks



10/4/2007

22

## Pentium 4 Misprediction Rate (per 1000 instructions, not per branch)

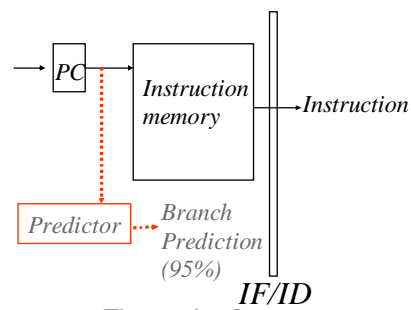


10/4/2007

23

## Branch Target Buffers (BTB)

- Branch prediction



Calculate the target address:  
 $(PC+4)$  or  $(PC+4+16\text{-bit offset} \ll 2)$

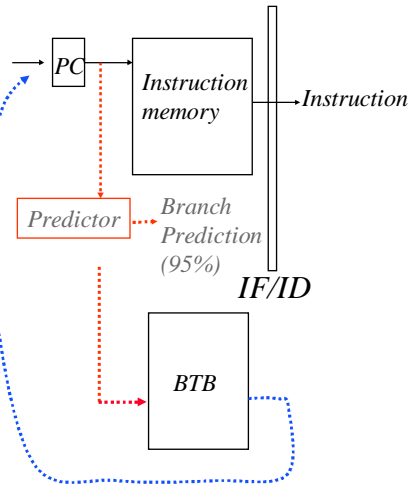
Branch target calculation is costly and stalls the instruction fetch.

10/4/2007

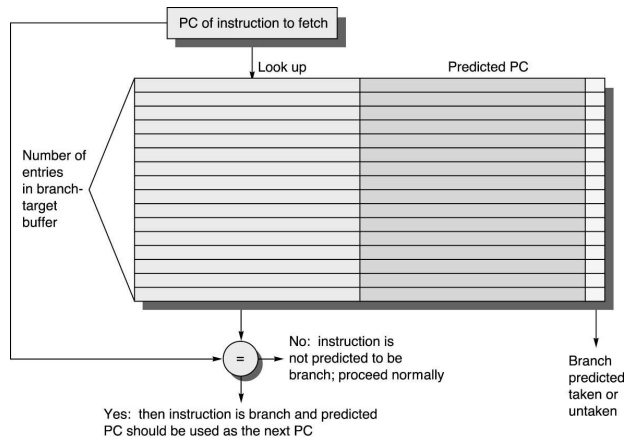
24

## Branch Target Buffers (BTB)

- BTB stores PCs the same way as caches
- The PC of a branch is sent to the BTB
- When a match is found the corresponding predicted PC is returned
- If the branch was predicted taken, instruction fetch continues at the returned predicted PC



## Branch Target Buffers (BTB)



© 2003 Elsevier Science (USA). All rights reserved.

## Dynamic Branch Prediction Summary

---

- **Prediction becoming important part of execution**
- **Branch History Table: 2 bits for loop accuracy**
- **Correlation: Recently executed branches correlated with next branch**
  - Either different branches (GA)
  - Or different executions of same branches (PA)
- **Tournament predictors take insight to next level, by using multiple predictors**
  - usually one based on global information and one based on local information, and combining them with a selector
  - In 2006, tournament predictors using  $\approx$  30K bits are in processors like the Power5 and Pentium 4
- **Branch Target Buffer: include branch address & prediction**

10/4/2007

27