CMSC 411
Computer Systems Architecture
Lecture 9
Instruction Level Parallelism 3
(Static & Dynamic Branch Prediction)

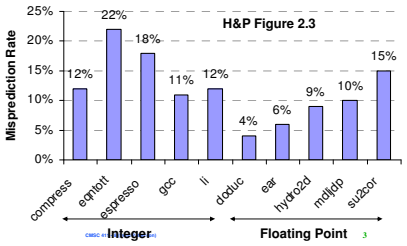
Outline

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

Static Branch Prediction

- Previously scheduled code around delayed branch
- To reorder code around branches
  - Need to predict branch statically during compile
- Simplest scheme is to predict a branch as taken
  - Average misprediction = untaken branch frequency = 34% SPEC92

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

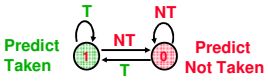


Dynamic Branch Prediction

- 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/compilers 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 that have dynamic behavior

Dynamic Branch Prediction

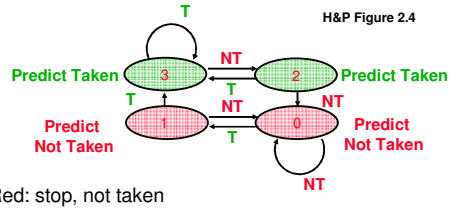
- Performance = f(accuracy, cost of misprediction)
- Branch History Table (BHT): table of 1-bit values indexed by lower bits of PC address index
  - Says whether or not branch taken last time
  - No address check (may refer to wrong branch)



- Problem: in a loop, 1-bit BHT will cause two mispredictions (avg is 9 loop iterations before exit):
  - End of loop, when it exits instead of looping as before
  - First time through loop on next time through code, when it predicts exit instead of looping

Dynamic Branch Prediction

- Solution: 2-bit prediction scheme where predictor changes prediction only if it mispredicts *twice* in a row

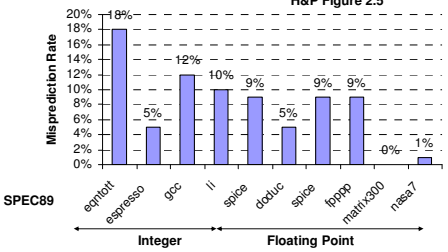


- Red: stop, not taken
- Green: go, taken
- Adds *hysteresis* to decision making process

BHT Accuracy

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

H&P Figure 2.5



Correlated Branch Prediction

- Idea – record m most recently executed branches as taken or not taken, and use that pattern to select the proper n-bit branch history table
- In general, (m,n) predictor means record last m branches to select between 2^m history tables, each with n-bit counters
  - Thus, old 2-bit BHT is a (0,2) predictor
  - Global Branch History: m-bit shift register keeping T/NT status of last m branches.
  - Each entry in table has 2^m n-bit predictors
- Also known as 2-level adaptive predictor

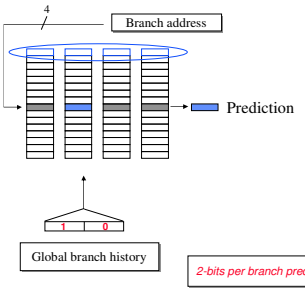
```
if (aa == 2)
    aa = 0;
if (bb == 2)
    bb = 0;
if (aa != bb) {
```

Depends on 2 previous branches!

Correlating Branches

(2,2) predictor w/

- Behavior of recent branches selects between four predictions of next branch, updating just that prediction

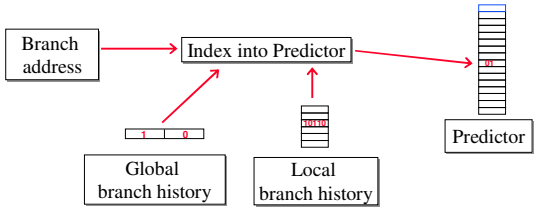


Or, 4 addr bits + 2 history bits give us 6-bit index into 2^6 = 64 predictors, each having two bits → 128 total bits.

2-bits per branch predictor

Correlated Branch Prediction

- Possible choices
  - Local history + branch address
  - Global branch history + branch address
  - Global branch history only (no branch address)
    - » Ignores branch instruction

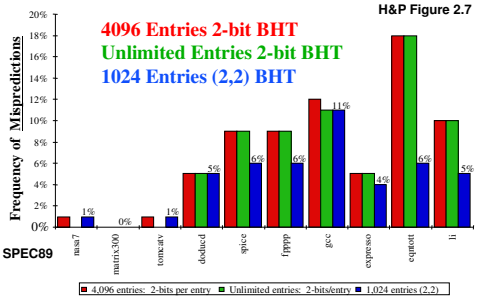


Calculations

- 4096-entry (0,2) predictor (i.e., 2-bit BHT)
  - 4k x 2 = 8k bits
  - 4k = 2^{12} → 12 address bits
- How to use the same # bits w/ a (2,2) predictor?
  - 8k bits w/ 2-bit BHT means 4k BHTs
  - the (2, 2) implies an entry has four BHTs
    - → 1k entries, i.e. a (2,2) predictor w/ 1024 entries

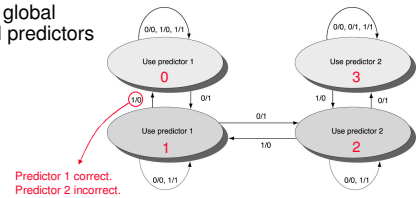
Accuracy of Different Schemes

H&P Figure 2.7



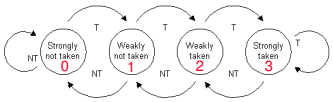
Tournament Predictors

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



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

N-bit Saturating Counter



T = taken
NT = not taken

- Used to choose between predictors X & Y
- N-bit counter value between 0 and 2^n-1
- Counter operations
  - Increment by 1 (up to 2^n-1)
    - » If X is correct & Y is incorrect
  - Decrement by 1 (down to 0)
    - » If Y is correct & X is incorrect
- Choose predictor X if counter > 2^{n-1}, Y otherwise
- Can be used as predictor (X = taken, Y = not taken)

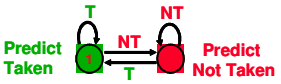
Tournament Predictor : DEC Alpha 21264

- 8K Tournament predictor using 4K 2-bit counters indexed by local branch address. Chooses between:
- Global predictor
  - 12 – 4K entries indexed by history of last 12 branches (2^{12} = 4K)
  - 8K – Each entry is a standard 2-bit predictor
- Local predictor
  - 10K – Local history table: 1K 10-bit entries recording last 10 branches, index by branch address
  - 3K – The pattern of the last 10 occurrences of that particular branch used to index table of 1K entries with 3-bit saturating counters

Total size of predictor = 8K + 8K + 10K + 3K = 29K

(0,1) Predictor

- Branches in loop
  - B1: BNEZ … // branch 1
  - B2: BNEZ … // branch 2
- Branch results
  - B1: T,NT,T,NT,T
  - B2: T,T,T,T,NT

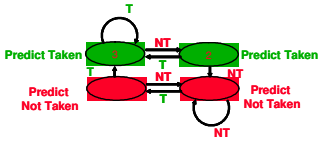


	Branch 1			Branch 2		
Iteration	Predictor	Prediction	Action	Predictor	Prediction	Action
1	0	NT	T	0	NT	T
2	1	T	NT	1	T	T
3	0	NT	T	1	T	T
4	1	T	NT	1	T	T
5	0	NT	T	1	T	NT
Exit loop	1			0		

Prediction based on state of predictor

(0,2) Predictor

- Branches in loop
  - B1: BNEZ … // branch 1
  - B2: BNEZ … // branch 2
- Branch results
  - B1: T,NT,T,NT,T
  - B2: T,T,T,T,NT



	Branch 1			Branch 2		
Iteration	Predictor	Prediction	Action	Predictor	Prediction	Action
1	0	NT	T	0	NT	T
2	1	NT	NT	1	NT	T
3	0	NT	T	3	T	T
4	1	NT	NT	3	T	T
5	0	NT	T	3	T	NT
Exit loop	1			2		

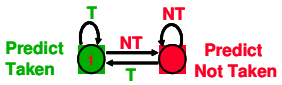
(0,2) Predictor w/ Saturating Counter

- Branches in loop
  - B1: BNEZ … // branch 1
  - B2: BNEZ … // branch 2
- Branch results
  - B1: T,NT,T,NT,T
  - B2: T,T,T,T,NT

	Branch 1			Branch 2		
Iteration	Predictor	Prediction	Action	Predictor	Prediction	Action
1	0	NT	T	0	NT	T
2	1	NT	NT	1	NT	T
3	0	NT	T	2	T	T
4	1	NT	NT	3	T	T
5	0	NT	T	3	T	NT
Exit loop	1			2		

(1,1) Predictor w/ Global History + Branch

- Branches in loop
  - B1: BNEZ … // branch 1
  - B2: BNEZ … // branch 2
- Branch results
  - B1: T,NT,T,NT,T
  - B2: T,T,T,T,NT



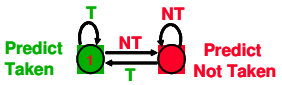
P0 / P1 → Last global branch Not taken / Taken

	Branch 1			Branch 2		
Iteration	Predictor	Prediction	Action	Predictor	Prediction	Action
1	? / ?		T	? / ?		T
2	? / ?		NT	? / ?		T
3	? / ?		T	? / ?		T
4	? / ?		NT	? / ?		T
5	? / ?		T	? / ?		NT
Exit loop	? / ?			? / ?		

Choose predictor based on last global branch action

(1,1) Predictor w/ Global History + Branch

- Branches in loop
  - B1: BNEZ … // branch 1
  - B2: BNEZ … // branch 2
- Branch results
  - B1: T,NT,T,NT,T
  - B2: T,T,T,T,NT

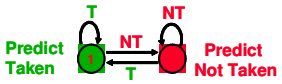


P0 / P1 → Last global branch Not taken / Taken

	Branch 1			Branch 2		
Iteration	Predictor	Prediction	Action	Predictor	Prediction	Action
1	0 / 0	NT	T	0 / 0	NT	T
2	1 / 0	NT	NT	0 / 1	NT	T
3	1 / 0	NT	T	1 / 1	T	T
4	1 / 1	T	NT	1 / 1	T	T
5	1 / 0	NT	T	1 / 1	T	NT
Exit loop	1 / 1			1 / 0		

(1,1) Predictor w/ Local History + Branch

- Branches in loop
  - B1: BNEZ … // branch 1
  - B2: BNEZ … // branch 2
- Branch results
  - B1: T,NT,T,NT,T
  - B2: T,T,T,T,NT



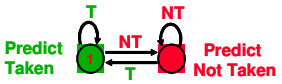
P0 / P1 → Last local branch Not taken / Taken

	Branch 1			Branch 2		
Iteration	Predictor	Prediction	Action	Predictor	Prediction	Action
1	? / ?		T	? / ?		T
2	? / ?		NT	? / ?		T
3	? / ?		T	? / ?		T
4	? / ?		NT	? / ?		T
5	? / ?		T	? / ?		NT
Exit loop	? / ?			? / ?		

Choose predictor based on last local branch action

(1,1) Predictor w/ Local History + Branch

- Branches in loop
  - B1: BNEZ … // branch 1
  - B2: BNEZ … // branch 2
- Branch results
  - B1: T,NT,T,NT,T
  - B2: T,T,T,T,NT

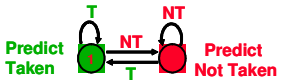


P0 / P1 → Last local branch Not taken / Taken

	Branch 1			Branch 2		
Iteration	Predictor	Prediction	Action	Predictor	Prediction	Action
1	0 / 0	NT	T	0 / 0	NT	T
2	1 / 0	NT	NT	1 / 0	NT	T
3	1 / 0	T	T	1 / 1	T	T
4	1 / 0	NT	NT	1 / 1	T	T
5	1 / 0	T	T	1 / 1	T	NT
Exit loop	1 / 0			1 / 0		

(2,1) Global Predictor (no Branch Addr)

- Branches in loop
  - B1: BNEZ … // branch 1
  - B2: BNEZ … // branch 2
- Branch results
  - B1: T,NT,T,NT,T
  - B2: T,T,T,T,NT



P0 / P1 / P2 / P3 → History = 00 / 01 / 10 / 11

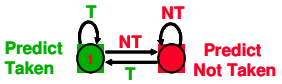
Branch actions stored in Global History

			Branch 1				Branch 2	
Iter	History	Predictor	Prediction	Action	History	Predictor	Prediction	Action
1	00	?/?/?/?	Same 4	T	01	?/?/?/?		T
2	11	?/?/?/?	Predictors!	NT	10	?/?/?/?		T
3	01	?/?/?/?		T	11	?/?/?/?		T
4	11	?/?/?/?		NT	10	?/?/?/?		T
5	01	?/?/?/?		T	11	?/?/?/?		NT
Exit	10							

History based on last 2 global branch actions; chose predictor based on history

(2,1) Global Predictor (no Branch Addr)

- Branches in loop
  - B1: BNEZ … // branch 1
  - B2: BNEZ … // branch 2
- Branch results
  - B1: T,NT,T,NT,T
  - B2: T,T,T,T,NT

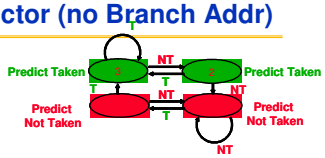


P0 / P1 / P2 / P3 → History = 00 / 01 / 10 / 11

			Branch 1				Branch 2	
Iter	History	Predictor	Prediction	Action	History	Predictor	Prediction	Action
1	00	0/0/0/0	NT	T	01	1/0/0/0	NT	T
2	11	1/1/0/0	NT	NT	10	1/1/0/0	NT	T
3	01	1/1/1/0	T	T	11	1/1/1/0	NT	T
4	11	1/1/1/1	T	NT	10	1/1/1/0	T	T
5	01	1/1/1/0	T	T	11	1/1/1/0	NT	NT
Exit	10	1/1/1/0						

(2,2) Global Predictor (no Branch Addr)

- Branches in loop
  - B1: BNEZ … // branch 1
  - B2: BNEZ … // branch 2
- Branch results
  - B1: T,NT,T,NT,T
  - B2: T,T,T,T,NT

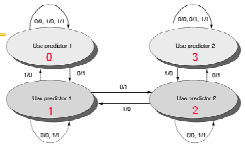


P0 / P1 / P2 / P3 → History = 00 / 01 / 10 / 11

			Branch 1				Branch 2	
Iter	History	Predictor	Prediction	Action	History	Predictor	Prediction	Action
1	00	0/0/0/0	NT	T	01	1/0/0/0	NT	T
2	11	1/1/0/0	NT	NT	10	1/1/0/0	NT	T
3	01	1/1/1/0	NT	T	11	1/3/1/0	NT	T
4	11	1/3/1/1	NT	NT	10	1/3/1/0	NT	T
5	01	1/3/3/0	T	T	11	1/3/3/0	NT	NT
Exit	10	1/3/3/0						

Tournament Predictor

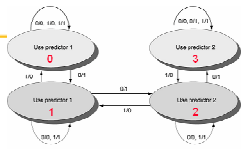
- 2-bit tournament predictor
  - Indexed by branch address
- Chooses between two predictors
  1. (2,2) Global Predictor
  2. (1,1) Predictor w/ Local History



	Branch 1					Branch 2				
Iter	2,2	1,1	Predictor	Predict	Action	2,2	1,1	Predictor	Predict	Action
1	NT	NT	0		T	NT	NT	0		T
2	NT	NT			NT	NT	NT			T
3	NT	T			T	NT	T			T
4	NT	NT			NT	NT	T			T
5	T	T			T	NT	T			NT
Exit										

Tournament Predictor

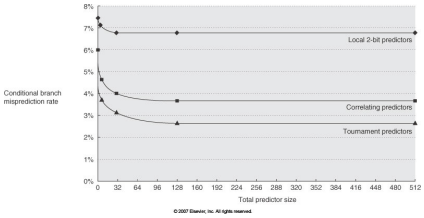
- 2-bit tournament predictor
  - Indexed by branch address
- Chooses between two predictors
  1. (2,2) Global Predictor
  2. (1,1) Predictor w/ Local History



	Branch 1					Branch 2				
Iter	2,2	1,1	Predictor	Predict	Action	2,2	1,1	Predictor	Predict	Action
1	NT	NT	0	NT	T	NT	NT	0	NT	T
2	NT	NT	0	NT	NT	NT	NT	0	NT	T
3	NT	T	0	NT	T	NT	T	0	NT	T
4	NT	NT	1	NT	NT	NT	T	1	NT	T
5	T	T	1	T	T	NT	T	2	T	NT
Exit			1					1		

Comparing Predictors (H&P Fig. 2.8)

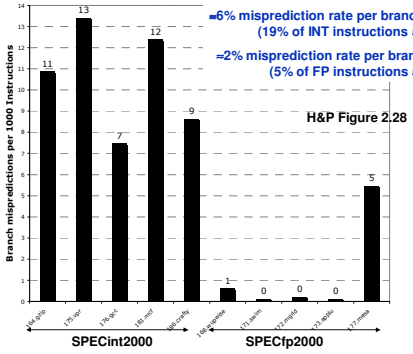
- Advantage of tournament predictor is ability to select the right predictor for a particular branch
  - Particularly crucial for integer benchmarks.
  - A typical tournament predictor will select the global predictor almost 40% of the time for the SPEC integer benchmarks and less than 15% of the time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



© 2007 Elsevier, Inc. All rights reserved.

28

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



≈6% misprediction rate per branch SPECint (19% of INT instructions are branch)

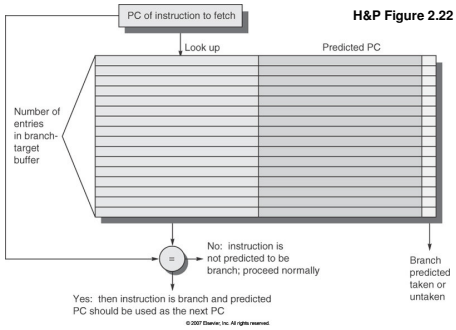
≈2% misprediction rate per branch SPECfp (5% of FP instructions are branch)

H&P Figure 2.28

Branch Target Buffers (BTB)

- Branch target calculation is costly and stalls the instruction fetch.
- 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



H&P Figure 2.22

© 2007 Elsevier, Inc. 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 ≈ 30K bits are in processors like the Power5 and Pentium 4
- Branch Target Buffer: include branch address & prediction