CS252
Graduate Computer Architecture

Lecture 18:
Branch Prediction + analysis resources => ILP

April 2, 2002

Prof. David E. Culler

Computer Science 252

Spring 2002

Today's Big Idea

- Reactive: past actions cause system to adapt use
  - do what you did before better
  - ex: caches
  - TCP windows
  - URL completion, ...
- Proactive: uses past actions to predict future actions
  - optimize speculatively, anticipate what you are about to do
  - branch prediction
  - long cache blocks
  - ???

Review: Case for Branch Prediction when Issue N instructions per clock cycle

1. Branches will arrive up to *n* times faster in an *n*-issue processor
2. Amdahl's Law => relative impact of the control stalls will be larger with the lower potential CPI in an *n*-issue processor

conversely, need branch prediction to 'see' potential parallelism

Review: 7 Branch Prediction Schemes

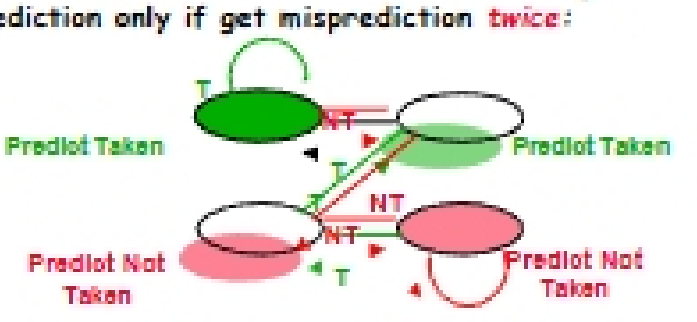
1. 1-bit Branch-Prediction Buffer
2. 2-bit Branch-Prediction Buffer
3. Correlating Branch Prediction Buffer
4. Tournament Branch Predictor
5. Branch Target Buffer
6. Integrated Instruction Fetch Units
7. Return Address Predictors

Review: Dynamic Branch Prediction

- Performance = *f*(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 (saves HW, but may not be right branch)
- Problem: in a loop, 1-bit BHT will cause 2 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
  - Only 80% accuracy even if loop 90% of the time

Review: Dynamic Branch Prediction
(Jim Smith, 1981)

- Better Solution: 2-bit scheme where change prediction only if get misprediction twice:



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

Consider 3 Scenarios

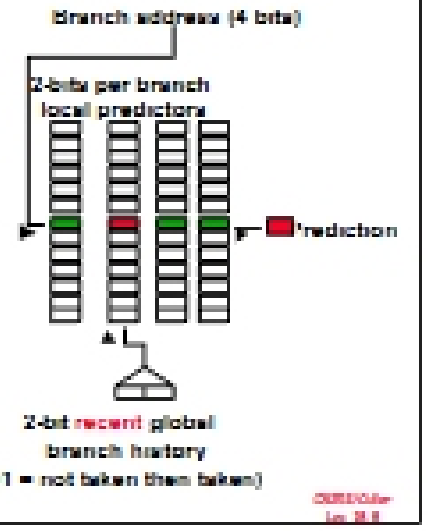
- Branch for loop test
- Check for error or exception
- Alternating taken / not-taken
  - example?
- Your worst-case prediction scenario

Correlating Branches

Idea: taken/not taken of recently executed branches is related to behavior of next branch (as well as the history of that branch behavior)

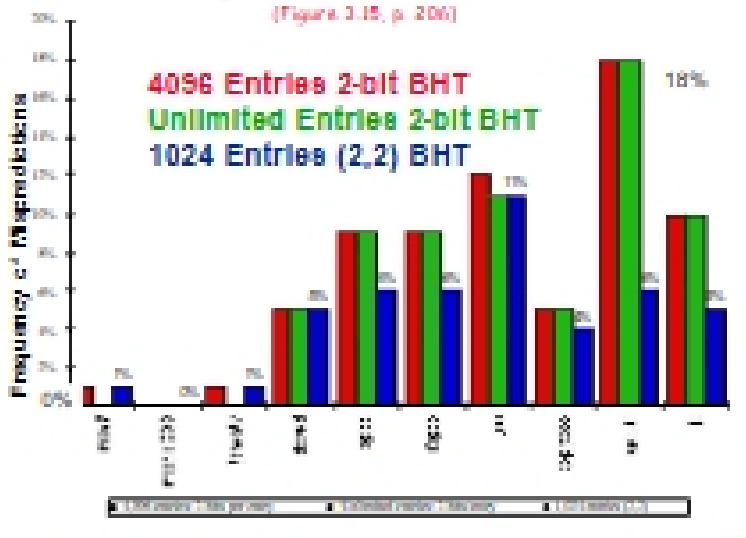
- Then behavior of recent branches selects between, say, 4 predictions of next branch, updating just that prediction

- (2,2) predictor: 2-bit global, 2-bit local



Accuracy of Different Schemes

(Figure 3.15, p. 206)



What's missing in this picture?

Re-evaluating Correlation

- Several of the SPEC benchmarks have less than a dozen branches responsible for 90% of taken branches:

program	branch %	static	# = 90%
compress	14%	236	13
eqntott	25%	494	5
gcc	15%	9531	2020
mpeg	10%	5598	532
real gcc	13%	17361	3214

- Real programs + OS more like gcc
- Small benefits beyond benchmarks for correlation? problems with branch aliases?

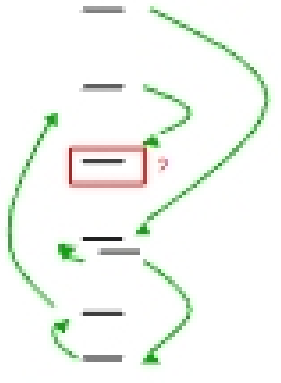
BHT Accuracy

- Mispredict because either:
  - Wrong guess for that branch
  - Got branch history of wrong branch when index the table
- 4096 entry table programs vary from 1% misprediction (nasa7, tomcatv) to 18% (eqntott), with spice at 9% and gcc at 12%
- For SPEC92, 4096 about as good as infinite table

Tournament Predictors

- Motivation for correlating branch predictors is 2-bit predictor failed on important branches; by adding global information, performance improved
- Tournament predictors: use 2 predictors, 1 based on global information and 1 based on local information, and combine with a selector
- Hopes to select right predictor for right branch (or right context of branch)

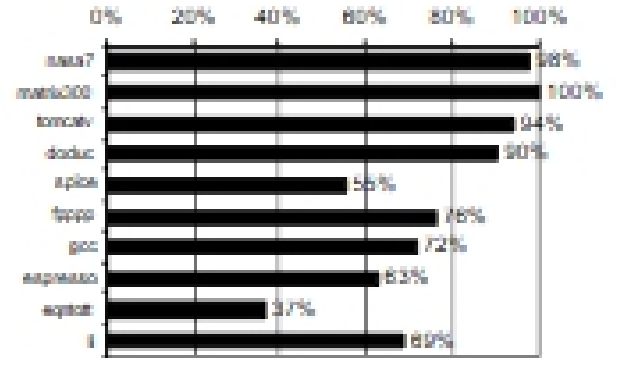
Dynamically finding structure in Spaghetti



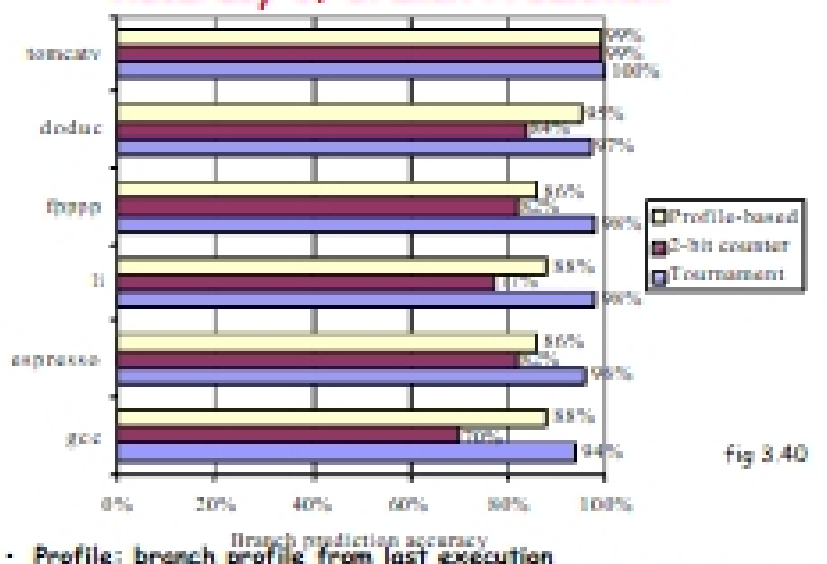
Tournament Predictor in Alpha 21264

- 4K 2-bit counters to choose from among a global predicter and a local predicter
- Global predicter also has 4K entries and is indexed by the history of the last 12 branches; each entry in the global predicter is a standard 2-bit predicter
  - 12-bit pattern: ith bit 0 => ith prior branch not taken; ith bit 1 => ith prior branch taken;
- Local predicter consists of a 2-level predicter:
  - Top level a local history table consisting of 1024 10-bit entries; each 10-bit entry corresponds to the most recent 10 branch outcomes for the entry. 10-bit history allows patterns 10 branches to be discovered and predicted.
  - Next level Selected entry from the local history table is used to index a table of 1K entries consisting a 3-bit saturating counters, which provide the local prediction
- Total size: 4K*2 + 4K*2 + 1K*10 + 1K*3 = 29K bits! (~180,000 transisters)

% of predictions from local predictor in Tournament Prediction Scheme

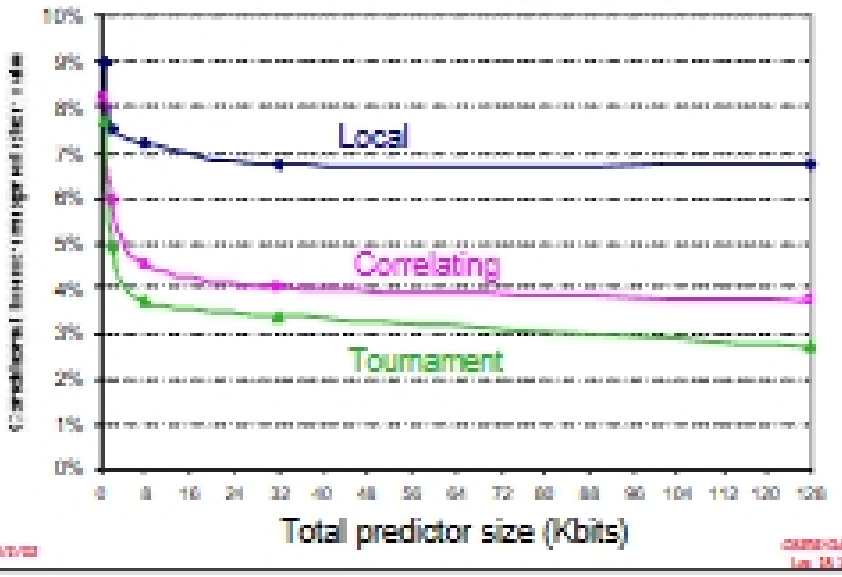


Accuracy of Branch Prediction



- Profile: branch profile from last execution (static in that it encoded in instruction, but profile)

Accuracy v. Size (SPEC89)



Need Address at Same Time as Prediction

- Branch Target Buffer (BTB): Address of branch index to get prediction AND branch address (if taken)
  - Note: must check for branch match now, since can't use wrong branch address (Figure 3.19, 3.20)