Branch Prediction, Multiple Issue

Four Branch Prediction Schemes

- 1-bit Branch-Prediction Buffer
- 2-bit Branch-Prediction Buffer
- Correlating Branch Prediction Buffer
- Tournament Branch Predictor
1-bit Branch Prediction

- **Branch History Table/Buffer:**
  - Indexed by low order bits of branch instruction address
  - Table of 1-bit values
  - Says whether branch taken (1) or not taken (0) last time
  - Use current value to predict; modify value if prediction turns out to be wrong.
- **Problems: in a loop, 1-bit BHT will cause 2 mispredictions:**
  - 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
  - Assuming 10 times thru loop (typical value) only 80% accuracy even if loop 90% of the time
  - No address check (may not be right branch stmt)

2-Bit Branch Prediction

(Jim Smith, 1981)

- Two bits of history -- must mispredict twice before changing the prediction
- Uses a 2-bit saturating counter

![2-bit branch prediction diagram]

- increment by 1
- decrement by 1
2-Bit Branch Prediction (cont.)

- **RED**: Predict not taken if counter < 2 (half of max = 4)
- **GREEN**: Predict taken if counter >= 2

Counter value is a history of last two branch outcomes

2-bit Branch Prediction

- **Branch History Table/Buffer**:
  - Indexed by low order bits of branch instruction address
  - Table of 2-bit values
  - Records history of last two branch outcomes
  - Use counter value to predict (NT if < 2, T if >= 2)

- **Performance**
  - Midprediction rate of 1 to 18% with 4K BHT
  - Requires more overhead than 1-bit (update history bit every time through the loop)

  - Note: error in Figure 3.7
N-bit Branch Prediction

- Generalization of 2-bit predictor using an n-bit saturating counter
- Predict not taken if counter < $2^{(n-1)}$
- Predict taken if counter $\geq 2^{(n-1)}$
Accuracy of 2-bit Prediction

(2,2) Correlating Branches

Idea: use two pieces of information: (1) global history of all recently executed branches is related to behavior of next branch and (2) local history of that specific branch.

- (2,2) predictor: 2-bit global, 2-bit local
Generalization:
(m,n) Correlating Branches

- m-bit global history
- n-bit local predictor which uses an n-bit saturating counter

Size of BHT:
- $2^k \times 2^m \times n$

Accuracy of Different Schemes

4096 Entries 2-bit BHT
Unlimited Entries 2-bit BHT
1024 Entries (2,2) BHT
Summary: 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
  - 2-bit or (2,2) correlating predictors good

Tournament Predictors

- Use 2 predictors, 1 based on global information and 1 based on local information, and choose using a selector
- Hopes to select right predictor for right branch
**Tournament Predictor in Alpha 21264**

- **Global predictor**: 4K entries and is indexed by the history of the last 12 branches; each entry in the global predictor is a standard 2-bit predictor.
  - 12-bit pattern: ith bit 0 => ith prior branch not taken; ith bit 1 => ith prior branch taken.

- **Local predictor** consists of a 2-level predictor:
  - **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**: $4K \times 2 + 4K \times 2 + 1K \times 10 + 1K \times 3 = 29K$ bits!
  ($\sim 180,000$ transistors)

---

**Transition labels** indicate outcome of last predictions:
P1 / P2 where
0 = wrong
1 = right

E.g. 1/0 means
P1 predicted right,
P2 predicted wrong
% of predictions from local predictor in Tournament Prediction Scheme

<table>
<thead>
<tr>
<th>Application</th>
<th>Local Predictor %</th>
</tr>
</thead>
<tbody>
<tr>
<td>nasa7</td>
<td>98%</td>
</tr>
<tr>
<td>matrix300</td>
<td>100%</td>
</tr>
<tr>
<td>tomcatv</td>
<td>94%</td>
</tr>
<tr>
<td>doduc</td>
<td>90%</td>
</tr>
<tr>
<td>spice</td>
<td>55%</td>
</tr>
<tr>
<td>fpppp</td>
<td>76%</td>
</tr>
<tr>
<td>gcc</td>
<td>72%</td>
</tr>
<tr>
<td>espresso</td>
<td>63%</td>
</tr>
<tr>
<td>eqntott</td>
<td>37%</td>
</tr>
<tr>
<td>li</td>
<td>69%</td>
</tr>
</tbody>
</table>

Accuracy of Branch Prediction

- **Profile-based**
- **2-bit counter**
- **Tournament**

**Profile:** branch profile from last execution
Recap: Branch Prediction Methods

- 1-bit
- 2-bit
- Correlating
- Tournament

Each is a subcase of the following, with tournament the most general.
New Techniques to Reduce Branch Penalty

- Branch Target Buffer
- Integrated Instruction Fetch Units
- Return Address Predictors

Branch Target Buffer: get target address during IF

- 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, p. 262)

![Diagram of Branch Target Buffer and prediction logic]

Extra prediction state bits
**Special Case Return Addresses**

- Register Indirect branch hard to predict address
- SPEC89 85% such branches for procedure return
- Since stack discipline for procedures, save return address in small buffer that acts like a stack: 8 to 16 entries has small miss rate

**Integrated IF Units**

- Integrated branch prediction: branch prediction built into IF
- Instruction prefetch
- Instruction memory access and buffering

We shall see that pipelining and caching interact closely.
Pitfall: Sometimes bigger and dumber is better

- 21264 uses tournament predictor (29 Kbits)
- Earlier 21164 uses a simple 2-bit predictor with 2K entries (or a total of 4 Kbits)
- SPEC95 benchmarks, 21264 outperforms
  - 21264 avg. 11.5 mispredictions per 1000 instructions
  - 21164 avg. 16.5 mispredictions per 1000 instructions
- Reversed for transaction processing (TP)!
  - 21264 avg. 17 mispredictions per 1000 instructions
  - 21164 avg. 15 mispredictions per 1000 instructions
- TP code much larger & 21164 hold 2X branch predictions based on local behavior (2K vs. 1K local predictor in the 21264)