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

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

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

Dynamic Branch Prediction
• 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:

<table>
<thead>
<tr>
<th>SPEC99</th>
<th>Integer</th>
<th>Floating Point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0%</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>5%</td>
<td>12%</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>9%</td>
</tr>
<tr>
<td></td>
<td>4%</td>
<td>9%</td>
</tr>
<tr>
<td></td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>4%</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>6%</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>8%</td>
<td>6%</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>12%</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>14%</td>
<td>12%</td>
</tr>
<tr>
<td></td>
<td>16%</td>
<td>14%</td>
</tr>
<tr>
<td></td>
<td>18%</td>
<td>16%</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>18%</td>
</tr>
</tbody>
</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) {
  if (aa == 2)
aa = 0;
  if (bb == 2)
bb = 0;
  if (aa != bb)
```

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

- Correlation Branches

  (2,2) predictor w/ (2,2)
  - 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.

- Global branch history
- Local branch history
- 4 bits per branch predictor

- Index into Predictor

- 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

- Frequency of Mispredictions

- 4096 Entries 2-bit BHT
- Unlimited Entries 2-bit BHT
- 1024 Entries (2,2) BHT

- Accuracy of Different Schemes
Tournament Predictors

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

N-bit Saturating Counter

- 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

- Tournament predictor using 4K 2-bit counters indexed by local branch address. Chooses between:
  - Global predictor
    - 12 entries indexed by history of last 12 branches ($2^{12} = 4K$)
  - Local predictor
    - 8K entries indexed by history of last 10 branches, index by branch address
    - Each entry is a standard 2-bit predictor
    - Local history table: 1K 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
- 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

- 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

(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
### (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

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Branch 1 Prediction</th>
<th>Action</th>
<th>Branch 2 Prediction</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 / ?</td>
<td>T</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>2 / ?</td>
<td>NT</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>2 / ?</td>
<td>NT</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>2 / ?</td>
<td>NT</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td>2 / ?</td>
<td>T</td>
<td>2 / ?</td>
<td>NT</td>
</tr>
<tr>
<td>Exit loop</td>
<td>2 / ?</td>
<td></td>
<td>2 / ?</td>
<td></td>
</tr>
</tbody>
</table>

Choose predictor based on last global 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

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Branch 1 Prediction</th>
<th>Action</th>
<th>Branch 2 Prediction</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 / ?</td>
<td>T</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>2 / ?</td>
<td>NT</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>2 / ?</td>
<td>NT</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>2 / ?</td>
<td>NT</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td>2 / ?</td>
<td>T</td>
<td>2 / ?</td>
<td>NT</td>
</tr>
<tr>
<td>Exit loop</td>
<td>2 / ?</td>
<td></td>
<td>2 / ?</td>
<td></td>
</tr>
</tbody>
</table>

Choose predictor based on last local branch action.

### (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

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Branch 1 Prediction</th>
<th>Action</th>
<th>Branch 2 Prediction</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 / ?</td>
<td>T</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>2 / ?</td>
<td>NT</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>2 / ?</td>
<td>NT</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>2 / ?</td>
<td>NT</td>
<td>2 / ?</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td>2 / ?</td>
<td>T</td>
<td>2 / ?</td>
<td>NT</td>
</tr>
<tr>
<td>Exit loop</td>
<td>2 / ?</td>
<td></td>
<td>2 / ?</td>
<td></td>
</tr>
</tbody>
</table>

History based on last 2 global branch actions; choose predictor based on history.
(2,2) Global Predictor (no Branch Addr)

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

<table>
<thead>
<tr>
<th>Bar</th>
<th>History</th>
<th>Branch 1</th>
<th>Action</th>
<th>History</th>
<th>Branch 2</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>NT</td>
<td>T</td>
<td>0</td>
<td>NT</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>1/1/1/0</td>
<td>NT</td>
<td>10</td>
<td>1/1/1/0</td>
<td>NT</td>
</tr>
<tr>
<td>3</td>
<td>01</td>
<td>1/1/1/0</td>
<td>NT</td>
<td>T</td>
<td>1/3/1/0</td>
<td>NT</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>1/3/1/1</td>
<td>NT</td>
<td>10</td>
<td>1/3/1/0</td>
<td>NT</td>
</tr>
<tr>
<td>5</td>
<td>01</td>
<td>1/2/3/0</td>
<td>T</td>
<td>T</td>
<td>1/3/3/0</td>
<td>NT</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bar</th>
<th>2.2</th>
<th>1.1</th>
<th>Predictor</th>
<th>Action</th>
<th>2.2</th>
<th>1.1</th>
<th>Predictor</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NT</td>
<td>NT</td>
<td>0</td>
<td>T</td>
<td>NT</td>
<td>NT</td>
<td>0</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>NT</td>
<td>NT</td>
<td>0</td>
<td>NT</td>
<td>NT</td>
<td>NT</td>
<td>0</td>
<td>NT</td>
</tr>
<tr>
<td>3</td>
<td>NT</td>
<td>NT</td>
<td>0</td>
<td>NT</td>
<td>NT</td>
<td>NT</td>
<td>0</td>
<td>NT</td>
</tr>
<tr>
<td>4</td>
<td>NT</td>
<td>NT</td>
<td>1</td>
<td>NT</td>
<td>NT</td>
<td>T</td>
<td>1</td>
<td>NT</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>T</td>
<td>1</td>
<td>T</td>
<td>NT</td>
<td>T</td>
<td>2</td>
<td>NT</td>
</tr>
<tr>
<td>Exit</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Branch 1</th>
<th>Branch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bar 2.2</td>
<td>2.2</td>
</tr>
<tr>
<td>1</td>
<td>NT</td>
</tr>
<tr>
<td>2</td>
<td>NT</td>
</tr>
<tr>
<td>3</td>
<td>NT</td>
</tr>
<tr>
<td>4</td>
<td>NT</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
</tr>
<tr>
<td>Exit</td>
<td>1</td>
</tr>
</tbody>
</table>

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
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