CMSC 611: Advanced Computer Architecture

Branch Prediction /
Cache
- **Prediction accuracy of a 4096-entry prediction buffer ranges from 82% to 99% for the SPEC89 benchmarks**

- **The performance impact depends on frequency of branching instructions and the penalty of misprediction**
If (aa == 2)  
   aa = 0;
If (bb == 2)  
   bb = 0;
If (aa != bb) {
   DSUBUI R3, R1, #2
   BNEZ  R3, L1 ; branch b1 (aa!=2)
   ANDI R1, R1, #0 ; aa=0
   L1:  SUBUI R3, R2, #2
        BNEZ  R3, L2 ; branch b2 (bb!=2)
        ANDI R2, R2, #0 ; bb=0
   L2:  SUBU R3, R1, R2 ; R3=aa-bb
        BEQZ R3, L3 ; branch b3 (aa==bb)
}

- The behavior of branch b3 is correlated with the behavior of b1 and b2
- Clearly of both branches b1 and b2 are untaken, then b3 will be taken
- A predictor that uses only the behavior of a single branch to predict the outcome of that branch can never capture this behavior
- Branch predictors that use the behavior of other branches to make a prediction are called correlating or two-level predictors

Hypothesis: recent branches are correlated; that is, behavior of recently executed branches affects prediction of current branch
(2,2) Correlating Predictors

- Record m most recently executed branches as taken or not taken, and use that pattern to select the proper branch history table
- (m,n) predictor means record last m branches to select between 2m history tables each with n-bit counters
  - Old 2-bit branch history table is a (0,2) predictor
- In a (2,2) predictor, the behavior of recent branches selects between, four predictions of next branch, updating just that prediction

Total size = $2^m \times n \times \#$ prediction entries selected by branch address
**Accuracy of Different Schemes**

- 4096 entries (2 bits/entry)
- Unlimited entries (2 bits/entry)
- 1024 entries (2,2)
if (d==0)
  d=1;
if (d==1)
  ...

L1:
  DSUBUI R3, R1, #1
  BNEZ R3, L2 ; branch b2 (d!=1)

L2:
  ...

• Assume that d has values 0, 1, or 2 (alternating between 0, 2 as we enter this segment)
• Assume that the sequence will be executed repeatedly
• Ignore all other branches including those causing the sequence to repeat
• All branches are initially predicted to untaken state
With a single bit predictor

NT = Not Taken (if condition is false)
T = Taken (if condition is true)

<table>
<thead>
<tr>
<th>d=?</th>
<th>b1 prediction</th>
<th>b1 action</th>
<th>New b1 prediction</th>
<th>b2 prediction</th>
<th>b2 action</th>
<th>New b2 prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>NT</td>
<td>T</td>
<td>T</td>
<td>NT</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>0</td>
<td>T</td>
<td>NT</td>
<td>NT</td>
<td>T</td>
<td>NT</td>
<td>NT</td>
</tr>
<tr>
<td>2</td>
<td>NT</td>
<td>T</td>
<td>T</td>
<td>NT</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>0</td>
<td>T</td>
<td>NT</td>
<td>NT</td>
<td>T</td>
<td>NT</td>
<td>NT</td>
</tr>
</tbody>
</table>

- All branches are mispredicted

if (d==0)
    d=1;
if (d==1)

 ....

L2:
if (d==0) 
  d=1;
if (d==1) 
  BNEZ R1, L1 ; branch b1 (d!=0)
  DADDI R1, R0, #1 ; d==0, sp d=1
L1:  DSUBUI R3, R1, #1
  BNEZ R3, L2 ; branch b2 (d!=1)
....
L2:
Tournament Predictors

- Multilevel branch predictors use several levels of branch prediction tables together with an algorithm to choose among them.
- Tournament selectors are the most popular form of multilevel branch predictors (e.g. DEC Alpha 21264).
- Tournament predictors combines both local and global predictor.
- Selection between the two predictors are based on a selector (2-bit counter).
- Make a transition with two wrong prediction using the current table for which the correct prediction would have been possible using the other predictor.

![Diagram of tournament predictors](image_url)
Performance of Tournament Predictors

Based on SPEC 89 benchmark

Tournament predictors slightly outperform correlating predictors
Cache & Memory

- Why do designers need to know about Memory technology?
  - Processor performance is usually limited by memory bandwidth
  - As IC densities increase, lots of memory will fit on chip
- What are the different types of memory?
- How to maximize memory performance with least cost?
**Problem:** Memory can be a bottleneck for processor performance

**Solution:** Rely on memory hierarchy of faster memory to bridge the gap
Memory Hierarchy

- **Temporal Locality** (Locality in Time):
  ⇒ Keep most recently accessed data items closer to the processor

- **Spatial Locality** (Locality in Space):
  ⇒ Move blocks consists of contiguous words to the faster levels

- **Speed**: Fastest
- **Size**: Smallest
- **Cost**: Highest

- **Compiler**

- **Hardware**

- **Operating System**

- **Secondary Storage (Disk)**