Split Last-Address Predictor

Enric Morancho, José María Llabería, Àngel Olivé
Departament d'Arquitectura de Computadors
Universitat Politècnica de Catalunya

Work supported by the Ministry of Education and Science of Spain (CICYT TIC-0429/95)
Introduction

• Processors must exploit more ILP

• To avoid restrictions imposed by true-data dependencies:
  - Prediction of the result of an operation
  - Speculative execution of dependent operations
Last-address predictor

• Loads: effective addresses prediction
• Prediction = last effective address computed by the load

• Captured Predictability = \(\frac{\text{correct predictions}}{\text{dynamic loads}} \times 100\)

• Maximum capturable predictability

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Pred. capt.</th>
</tr>
</thead>
<tbody>
<tr>
<td>go</td>
<td>52.7</td>
</tr>
<tr>
<td>li</td>
<td>35.6</td>
</tr>
<tr>
<td>ijpeg</td>
<td>21.7</td>
</tr>
<tr>
<td>perl</td>
<td>79.2</td>
</tr>
</tbody>
</table>
Unified Predictor (UP)

• Update policy: always update

- Counters prevent some mispredictions (accuracy)
  - Also prevent some correct predictions
  - Unpredictable loads can evict predictable ones
Predictability distribution

- Predictability is distributed non-uniformly

- Loads highly predictable
- Loads highly unpredictable

Diagram showing predictability distribution across dynamic load instructions for different programs (go, vortex, gcc).
Idea of the Split Predictor (SP)

- Filtering load instructions that update prediction table
  - Only predictable loads update table
    - ✓ table size reduction
  - Loads must be classified

![Diagram showing the classification of addresses]

- Classification Table (CT)
- Address Table (AT)
- Always Update
- Only predictable
Miss rate

- Benchmark go, perfect filtering, LRU replacement policy

- Miss-rate reduction
- Similar miss rate: UP 1024 entries, SP 512 entries
Classifying mechanism $<N,k>$

- Uses $N$ bits discarding the $k$ low-order bits ($<N,k>$)

- Similarity of $<3,3>$ is 80%
Split Predictor implementation

- CT classifies load instructions using mechanism <3,3>
- SP places in AT only predictable load instructions

- Direct mapping
- Ratio CT entries and AT entries
Captured predictability

- Benchmark go, both predictors, ratio $\frac{\text{CTentries}}{\text{ATentries}} = 8$

Graph showing captured predictability and area cost for different entries.
Accuracy

- Mispredictions can produce a performance penalty

- *Accuracy* = \( \frac{\text{correct predictions}}{\text{predictions performed}} \times 100 \)
Conclusions

• Effective addresses are predictable

• Predictability is not distributed uniformly

• Updates considering predictability

• With 19% area-cost reduction, the SP maintains captured predictability and increases accuracy of the UP

• The same idea can be applied to value prediction and stride-based prediction