Alternate Path Fetch

Aniket Deshmukh, Chester Cai, Yale Patt

HPS Research Group
Motivation

• Branch mispredictions still limit single-thread performance.
Motivation

• Branch mispredictions still limit single-thread performance.

• Investing into more predictor capacity is providing diminishing returns.

• Academia has proposed alternatives to branch prediction, but these generally have low misprediction coverage.

Very few solutions that work for any conditional branch misprediction.
Branch Misprediction Penalty

Re-fill delay = frontend latency, a third of the total misprediction penalty
Alternate Path Fetch

- Branch A correctly predicted
  - Restore contents of pipeline registers, PC and Branch History

- Branch A mispredicted
  - Predicted Path (X)
  - Alternate Path (X)
Key Design Decisions

• For how long do we process instructions on the Alternate Path?
  • Stop at RAT accesses!

• Which branches do we pick for APF?
  • Low TAGE confidence + custom H2P table in conjunction to prioritize branches more likely to mispredict and get good coverage.

• Predicting branches on the Alternate Path?
  • Sharing bad, so we bank frontend structures.
Banking TAGE

Alternate Path
PC + History

Predicted Path
PC + History

pred_pc_hash==00

1

TAGE 16KB

pred_pc_hash==01

1

TAGE 16KB

pred_pc_hash==10

1

TAGE 16KB

pred_pc_hash==11

1

TAGE 16KB

alt_pc_hash

Alternate Path Prediction

11

pred_pc_hash

11

Predicted Path Prediction

11
Effect of Banking TAGE on **Baseline**
Overview of APF Specifications

- Implemented APF microarchitecture in Scarab\(^1\), an execution-driven x86 simulator.
- Used SPECINT2017, GAP for evaluation.

| Core | 3.2GHz, 8-wide issue, TAGE-SC-L Predictor [37]  
512 Entry ROB, 256 Entry Reservation Station, 16-wide retire  
15-cycle frontend latency, Decoupled BP  
192 entry load queue, 128 entry store queue |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Ports</td>
<td>6 ALU (3 can handle BRs, 3 can handle FP), 3 Load, 3 Store</td>
</tr>
</tbody>
</table>
| Predictors | TAGE-SC-L for condition branches  
History-based indirect branch predictor, RAS |
| Caches | 32KB 8-way L1 I-cache (4-cycle access)  
48KB 12-way D-cache (4-cycle access)  
1MB 16-way LLC cache (18-cycle access), 64B lines |
| Memory | DDR4_2400R: 1 rank, 2 channels  
4 bank groups and 4 banks per channel  
tRP-tCL-tRCD: 16-16-16 |
| Alternate Path Fetch | H2P Table: 128-entry, 0.1KB, 1-cycle access  
Alternate Path Buffers (3.2KB), 4 Buffers |
| APF Pipeline 13-cycle latency | BP, BTB, and I-Cache banking,  
Fetch and Decode Stages  
Dependency check logic (Pre-Rename) |

\(^1\)https://github.com/hpsresearchgroup/scarab
Performance

![Graph showing the performance metrics for different benchmarks. The graphs display the number of mispredictions per 1k instructions and percentage speedup for various benchmarks including perlbench, gcc, mcf, omnetpp, xalancbmk, x264, deepsjeng, leela, exchange2, xz, bfs, sssp, pr, cc, bc, tc, and gmean.](image-url)
Conclusion

• APF provides a relatively simple solution that helps reduce the misprediction penalty for any conditional branch.

• Because the alternate path fetch cycles can be flexibly distributed, APF provides additive benefits when combined with other specialized solutions.

• The TAGE banking scheme provides a simple mechanism to predict branches for two distinct instruction streams simultaneously.
Thank you

Aniket Deshmukh, Chester Cai, Yale Patt

HPS Research Group
Backup slides

Aniket Deshmukh, Chester Cai, Yale Patt
HPS Research Group
Overall Implementation

- Stop before RAT access!
- Addition port to I-Cache, BP and BTB or share frontend cycles between the alternate path and predicted path?
Scheduling fetch cycles for APF branches

Accuracy: 80%

P(Pred_A) = 0.80  P(Alt_A) = 0.20

Accuracy: 80%

P(Pred_B) = 0.64  P(Alt_B) = 0.16

Pick oldest first using TAGE confidence, pick oldest first using H2P Table.

<table>
<thead>
<tr>
<th></th>
<th>Coverage (Specificity)</th>
<th>Wastage (1-PVN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2P Table</td>
<td>95.43%</td>
<td>89.61%</td>
</tr>
<tr>
<td>TAGE confidence</td>
<td>56.33%</td>
<td>74.52%</td>
</tr>
</tbody>
</table>

**TABLE II**
Mispredictions detected by H2P Table and TAGE confidence
Reducing Pipeline Re-fill Delay

![Graph showing the impact of re-fill delay reduction on IPC relative to baseline for various benchmarks.]
Misprediction coverage
Time-sharing vs additional ports

![Bar chart comparing different benchmark programs with different methods: APF + Time-sharing, APF + Banking, and APF + Two ports. The x-axis represents different benchmarks, and the y-axis represents percentage speedup.](chart.png)
APF Buffer, Core size sweeps

[Graph showing percentage speedup with buffer size sweeps and core size sweeps.]
Comparison

- Misprediction Recovery Caches (no alternate path Predictions)
- Detecting Dependencies between instructions within a fetch packet
- Increased execution pressure
- RAT access
- Assign ROB, RS, etc
- Alternate Path Fetch
- Dual Path Instruction Processing
- Predication, Control Independence
Sweeping APF latency