Scalable Instruction Fetch and Trace Caching

Prof. Scott Rixner
Duncan Hall 3028
rixner@rice.edu

January 24, 2006
Instruction Stream Characteristics

- Small basic blocks
  - 4-5 instruction for integer applications
- Temporal locality
  - Instructions are frequently re-executed “soon”
- Branch behavior
  - Most branches are strongly biased or follow a pattern
Instruction Fetch Obstacles

- **Instruction cache misses**
  - Cannot fetch an instruction that is not there

- **Branches**
  - Cannot fetch an instruction until you know which one to fetch

- **What else?**
Branch Target Prediction

- **Branch Target Buffer**
  - Accessed during branch prediction
  - Holds predicted branch target addresses (branch target location from previous executions)
  - If branch is predicted taken, then next fetch is to the predicted target

- **Basic Block Target Buffer**
  - Also includes length of basic block, taken and fall-through addresses
  - Branch predictor selects next fetch (after sequential fetches have completed) from taken or fall-through addresses
Scalable Instruction Fetch

- Expense, in terms of delay, of building a better branch predictor and/or instruction prefetcher?

- Interconnect scaling
  - Delay becoming significantly worse
  - Seriously affects memory structures

- Scalable fetch target buffer
  - Two-level memory structure
  - Decoupling using fetch target queue
Scalable Instruction Fetch

- **Objectives**
  - Perform a useful prediction each cycle
  - Maximize the number of instructions fetched per prediction

- Are these good/the best objectives?

- How do they achieve scalability?
Decoupled Front End

- Branch predictor independent of pipeline
  - Predictions proceed ahead of instruction fetch stage
  - FTB used to predict the address and size of fetch blocks

- Instruction cache
  - Prefetches initiated by FTQ entries
  - Instructions fetched based on supplied predictions

- Extra state required
  - Speculative history queue
  - Speculative return address stack
IPC for BBTB and Single-level FTB

- Reinman, Austin, & Calder 5/99
Instructions/ns for FTB in 0.1μm

- Reinman, Austin, & Calder 5/99
Predicting Multiple Branches

-Rottenberg, Bennett, & Smith 1996
Trace Cache

- **Objectives**
  - Predict multiple branches per cycle
  - Fetch multiple basic blocks per cycle
  - Use normal instruction cache when traces are not available

- Are these good/the best objectives?

- Is this scalable?
# Basic Blocks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>taken %</th>
<th>avg basic block size</th>
<th># instr between taken branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>eqntott</td>
<td>86.2%</td>
<td>4.20</td>
<td>4.87</td>
</tr>
<tr>
<td>espresso</td>
<td>63.8%</td>
<td>4.24</td>
<td>6.65</td>
</tr>
<tr>
<td>xlisp</td>
<td>64.7%</td>
<td>4.34</td>
<td>6.70</td>
</tr>
<tr>
<td>gcc</td>
<td>67.6%</td>
<td>4.65</td>
<td>6.88</td>
</tr>
<tr>
<td>sc</td>
<td>70.2%</td>
<td>4.71</td>
<td>6.71</td>
</tr>
<tr>
<td>compress</td>
<td>60.9%</td>
<td>5.39</td>
<td>8.85</td>
</tr>
</tbody>
</table>

-Rottenberg, Bennett, & Smith 1996
Branch Address Cache

- Rottenberg, Bennett, & Smith 1996
Collapsing Buffer

-Rottenberg, Bennett, & Smith 1996
Trace Cache Performance

IPC for the Various Fetch Mechanisms, Single-Cycle Latency

SPEC92 benchmark

- Rottenberg, Bennett, & Smith 1996
Latency Effects

Performance Improvement over SEQ.3, Non-unit Latency

-Rottenberg, Bennett, & Smith 1996
Trace Cache Efficiency

Comparing Trace Cache to Ideal

- Rottenberg, Bennett, & Smith 1996
Instruction Fetch Obstacles

- Instruction cache misses
- Branches
  - Misprediction
  - Taken branches disrupt sequential control flow
  - Throughput
- Fetch unit latency
Next Class

Thursday – Instruction-level Parallelism

– “Instruction Issue Logic for Pipelined Supercomputers”
– “Implementation of Precise Interrupts in Pipelined Processors”