Link-Time Optimization for Instruction Cache Power Efficiency

Timothy M. Jones
Computer Laboratory
University of Cambridge
Outline

- Instruction cache power usage
  - Tagless instruction caching
    - Link-time optimisation
    - Hardware modifications
- Evaluation
- Conclusions
The Energy Problem

“We will soon spend more energy moving information than moving actual goods.”
Prith Banerjee, Director of HP Labs. Keynote at HPCA 2009

*Carbon cost* of Google revealed

Two search requests on the internet website Google produce "as much carbon dioxide as boiling a kettle", according to a Harvard University academic.

US physicist Alex Wissner-Gross claims that a typical Google search on a desktop computer produces

- Big impact tackling processor power consumption
- Using the compiler is novel & has large potential
Where Can The Compiler Help?

- Compiler's primary job is to create binaries
- It knows most about instructions
- The instruction cache is a good target
  - Frequently accessed
  - A hotspot
  - Requires high performance
Normal Instruction Cache Access

Address mapping for normal instruction cache access.

- **TAG**: Tag field for address containment.
- **INSN**: Instruction field containing the instruction.
- **way 0** and **way 1**: Two ways for accessing the cache.
- **normal access**: Access to the cache for instructions.

Diagram showing address mapping with two ways (way 0 and way 1) for normal instruction cache access.
All Tag and Insn Banks Accessed

way 0

TAG

INSN

way 1

TAG

INSN

normal access

address
Tag Checks Performed

- Address
- Tag
- Normal access
- Miss
- Hit
- Way 0
- Way 1

Tag

TAG

INSN

TAG

INSN

Timothy M Jones
Instruction Read Out

- TAG
- INSN

==

way 0 miss en

TAG

way 1 hit en

TAG

normal access

insn out

address
Currently Wasteful If Instruction Location is Known

<table>
<thead>
<tr>
<th>TAG</th>
<th>INSN</th>
<th>TAG</th>
<th>INSN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We Can Do Better: Way Prediction
We Can Do Better: Way Prediction
We Can Do Better: Way Prediction
Even Better: Tagless Accessing
Cache Behaves Like A Scratchpad
Best of Both Worlds
Tagless Access

way 0

way 1

address

tagless access

TAG

INSN

TAG

INSN

TAG

INSN
Only One Bank Accessed
No Tag Checks Required
Instruction Read Out

TAG

==

TAG

==

address

way 0

==

en

way 1

==

en

tagless access

insn out
Tagless Accessing

- Use cache like a scratchpad when possible
  - For power saving

- Use like a normal cache at other times
  - When flexibility is required

- Requires careful management
  - Performance losses otherwise
Outline

- Instruction cache power usage

- Tagless instruction caching
  - Link-time optimisation
  - Hardware modifications

- Evaluation

- Conclusions
Two Aspects

<table>
<thead>
<tr>
<th>VPA</th>
<th>PPA</th>
<th>T</th>
<th>Rway</th>
<th>R</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPA</td>
<td>PPA</td>
<td>T</td>
<td>Rway</td>
<td>R</td>
<td>V</td>
</tr>
<tr>
<td>VPA</td>
<td>PPA</td>
<td>T</td>
<td>Rway</td>
<td>R</td>
<td>V</td>
</tr>
<tr>
<td>VPA</td>
<td>PPA</td>
<td>T</td>
<td>Rway</td>
<td>R</td>
<td>V</td>
</tr>
</tbody>
</table>
Where Can The Compiler Help?

![Graph showing the relationship between percentage of time and instruction window size.]

- **X-axis:** Instruction Window Size
- **Y-axis:** Percentage of Time
Compiler Algorithm

- Our scheme uses new code placement algorithm
  - Converts temporal locality to spatial locality
  - Group frequently executed code into regions
    - Access without tag checks
    - Others accessed normally (with tag checks)

- Algorithm goals
  - Spend as much time as possible in tagless regions
  - Minimise switching between regions
  - Fill regions as much as possible
Example - Algorithm

Original binary used as input
Example - Algorithm

Blocks sorted by execution frequency
Example - Algorithm

Control flow graph constructed
Example - Algorithm

Initial clusters created
Example - Algorithm

Clusters expanded where profitable
Example - Algorithm

Best cluster selected and grown again
Link-Time Optimisation Summary

- Spatial locality from temporal
- Hot basic blocks in regions
  - No tag checks
- Maximise region code
- Minimise region switching
Additional Hardware Support

- Tagless regions identified in the ITLB

- Each page descriptor records information
  - Whether tagless or not
  - Which region of the cache the page is mapped to

- ITLB maintains LRU chains for cache regions
  - Used when a new tagless page is brought into the TLB
  - Prevents conflicts between regions
ITLB Support

<table>
<thead>
<tr>
<th>ITLB</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>VPA</td>
<td>PPA</td>
</tr>
<tr>
<td>VPA</td>
<td>PPA</td>
</tr>
<tr>
<td>VPA</td>
<td>PPA</td>
</tr>
<tr>
<td>VPA</td>
<td>PPA</td>
</tr>
</tbody>
</table>

**Diagram:**

- **TAG**
- **insn**
- **address**
- **way 0**
- **en**
- **en**
- **way 1**
- **insn out**
- **tagless access**
## ITLB Support

Whether a page is tagless

<table>
<thead>
<tr>
<th>VPA</th>
<th>PPA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPA</td>
<td>PPA</td>
<td>T</td>
</tr>
<tr>
<td>VPA</td>
<td>PPA</td>
<td>T</td>
</tr>
<tr>
<td>VPA</td>
<td>PPA</td>
<td>T</td>
</tr>
</tbody>
</table>

![Diagram of ITLB Support]

- **ITLB**: A component of the memory hierarchy responsible for translating virtual addresses to physical addresses. It supports tagless access.
- **VPA** (Virtual Page Address): The virtual address of the page.
- **PPA** (Physical Page Address): The physical address allocated to the page.
- **T**: Indicates whether a page is tagless (T for tagless, F for tagged).

The diagram shows a flow of data from the address to the ITLB, through ways 0 and 1, and eventually to the INSN (Instruction) page. The tagless access is highlighted, showing the path where no tag is required for access.
ITLB Support

The way in the cache
ITLB Support

The region in the cache
ITLB Support

Prevent conflicts in ITLB

Speculate

ITLB

<table>
<thead>
<tr>
<th>VPA</th>
<th>PPA</th>
<th>T</th>
<th>Rway</th>
<th>R</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPA</td>
<td>PPA</td>
<td>T</td>
<td>Rway</td>
<td>R</td>
<td>V</td>
</tr>
<tr>
<td>VPA</td>
<td>PPA</td>
<td>T</td>
<td>Rway</td>
<td>R</td>
<td>V</td>
</tr>
<tr>
<td>VPA</td>
<td>PPA</td>
<td>T</td>
<td>Rway</td>
<td>R</td>
<td>V</td>
</tr>
</tbody>
</table>
Instruction Cache Support

Tagless valid bits mark lines as valid in tagless mode
Instruction Cache Support

Tagless and tagged instructions can coexist
Instruction Cache Support

Conflicts reset the region's tv-bits
Hardware Summary

- Mostly ITLB changes
  - Whether a page is tagless
  - Cache way
  - Cache region

- Tagless valid bits in cache
  - Mix tagged and tagless data

- Speculate ITLB information
Outline

- Instruction cache power usage
- Tagless instruction caching
  - Link-time optimisation
  - Hardware modifications
- Evaluation
- Conclusions
Evaluation

- Compiler pass in Diablo link-time optimiser
- Spec Integer benchmarks
- High performance system simulated
  - Intel Core type processor (OoO superscalar and SMT)
- Three comparison schemes
  - Hardware (no link-time layout)
  - Layout only (no hardware support)
  - Tagless (both link-time layout and hardware support)
Does Not Slow Processor Down

![Graph showing normalized execution time for different processors. The x-axis represents different programs (bzip2, crafty, eon, gap, gcc, gzip, mcf, parser, perlbench, twolf, vortex, vpr, amean), and the y-axis represents normalized execution time (%). The graph compares Hardware, Layout Only, and Tagless processors. The Hardware and Layout Only processors show similar execution times, while the Tagless processors have slightly higher execution times.](image-url)
And Large Power Savings (49%)
Power Savings with SMT (41%)
Conclusions

- **Tagless access to instruction cache**
  - New linker code layout algorithm
    - Places frequently-executed code into regions
  - Mostly ITLB hardware support
    - Keeps track of mappings into the cache

- **Evaluation shows benefits of joint approach**
  - No performance loss
  - Large power savings
  - Savings maintained with SMT too
Thank You!

Collaborators: Jonas Maebe, Sandro Bartolini and Dominique Chanet

Link-Time Optimization for Power Efficiency in a Tagless Instruction Cache, CGO 2011