Link-Time Optimization for Instruction Cache Power Efficiency

Timothy M. Jones

Computer Laboratory
University of Cambridge

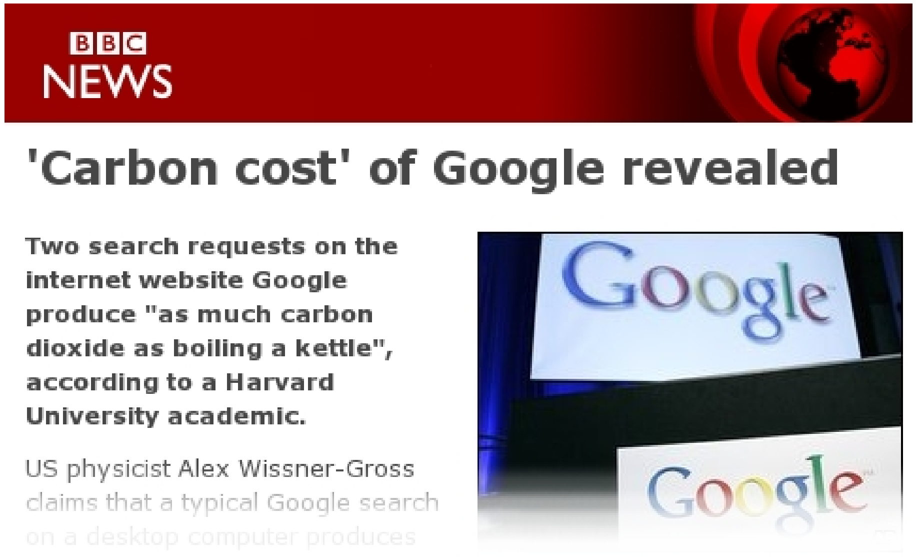
Outline

- **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

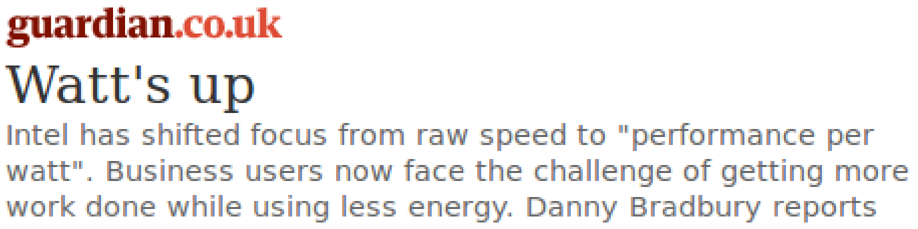


BBC NEWS

'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



guardian.co.uk

Watt's up

Intel has shifted focus from raw speed to "performance per watt". Business users now face the challenge of getting more work done while using less energy. Danny Bradbury reports

ARM THE ARCHITECTURE FOR THE DIGITAL WORLD

29 September 2008

ARM, Chartered, IBM and Samsung Collaborate to Enable Energy-Efficient 32nm and 28nm Systems On Chip

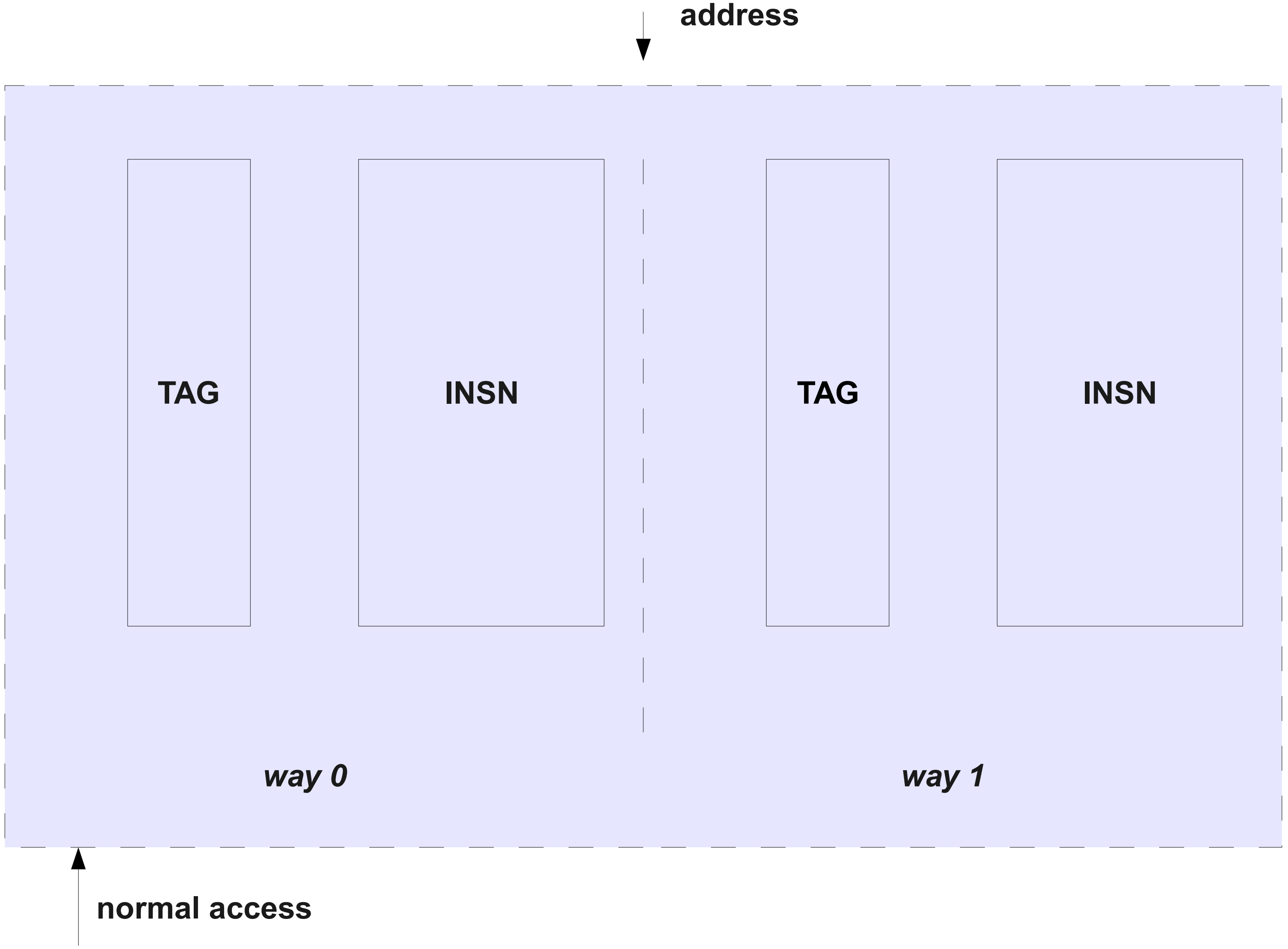
ARMto develop and license comprehensive Physical IP Design Platform targeted at achieving optimal power, performance, and area for current and future ARM Cortex processors.

- **Big impact tackling processor power consumption**
- **Using the compiler is novel & has large potential**

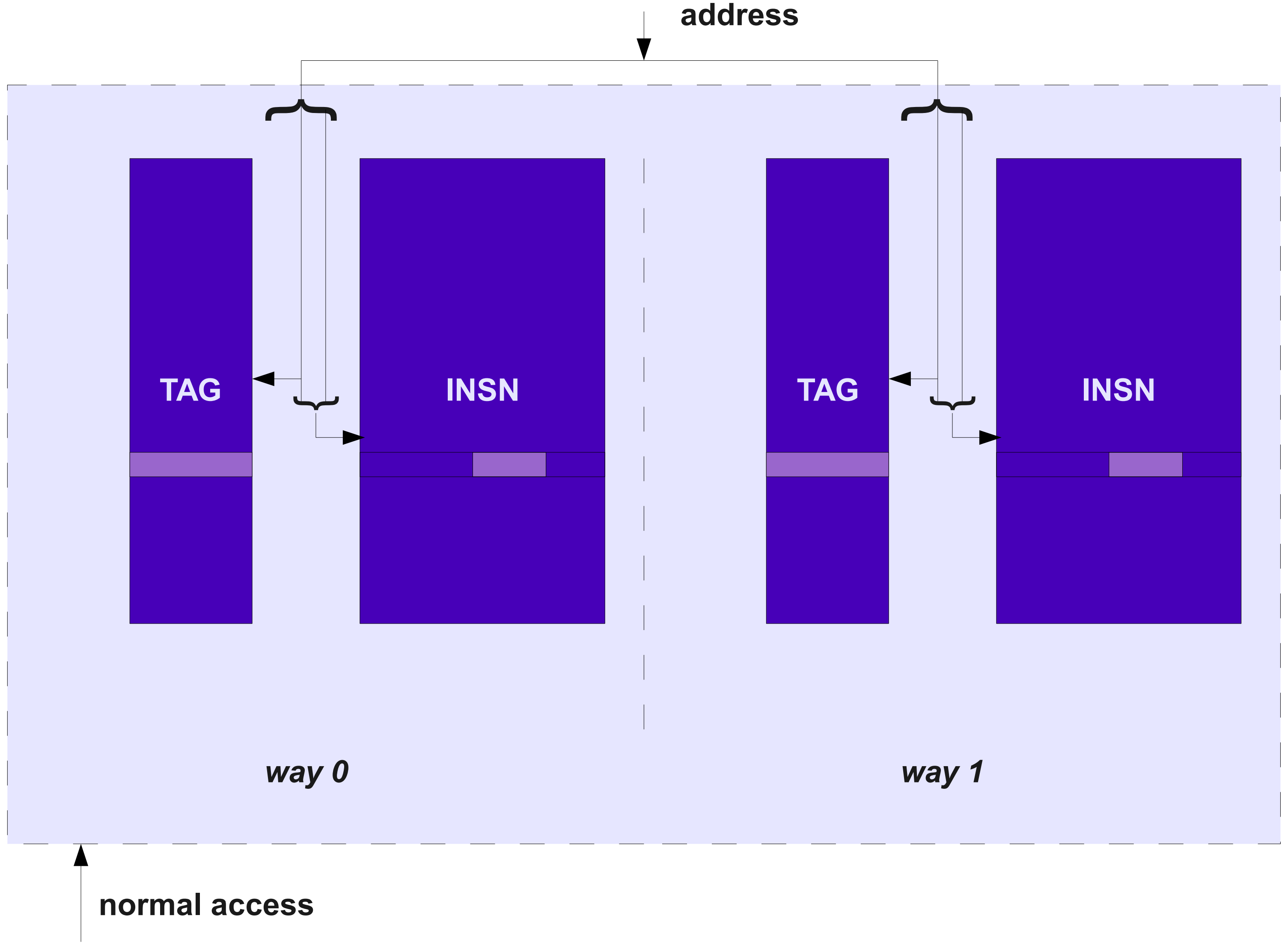
Where Can The Compiler Help?

- **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

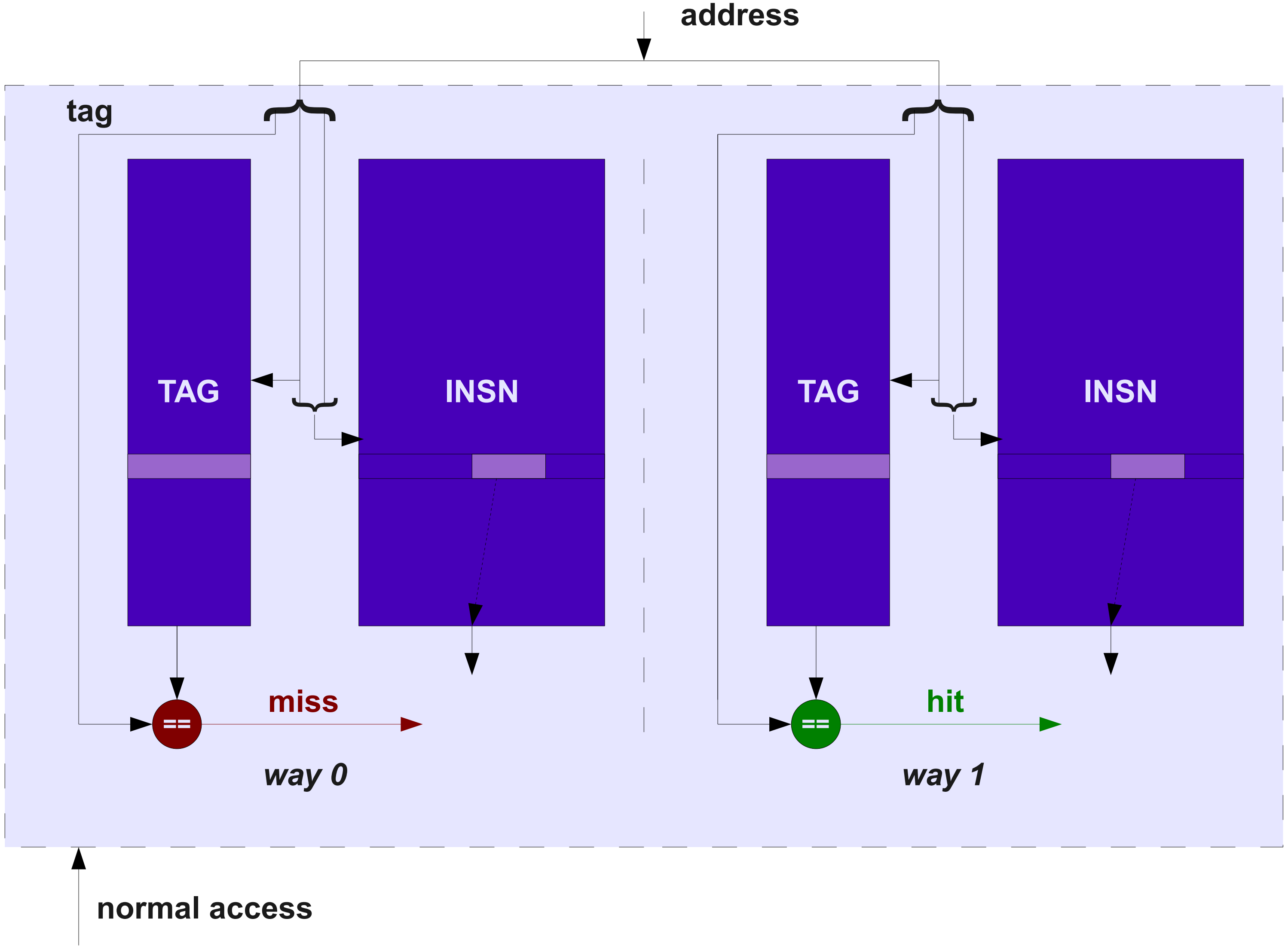
Normal Instruction Cache Access



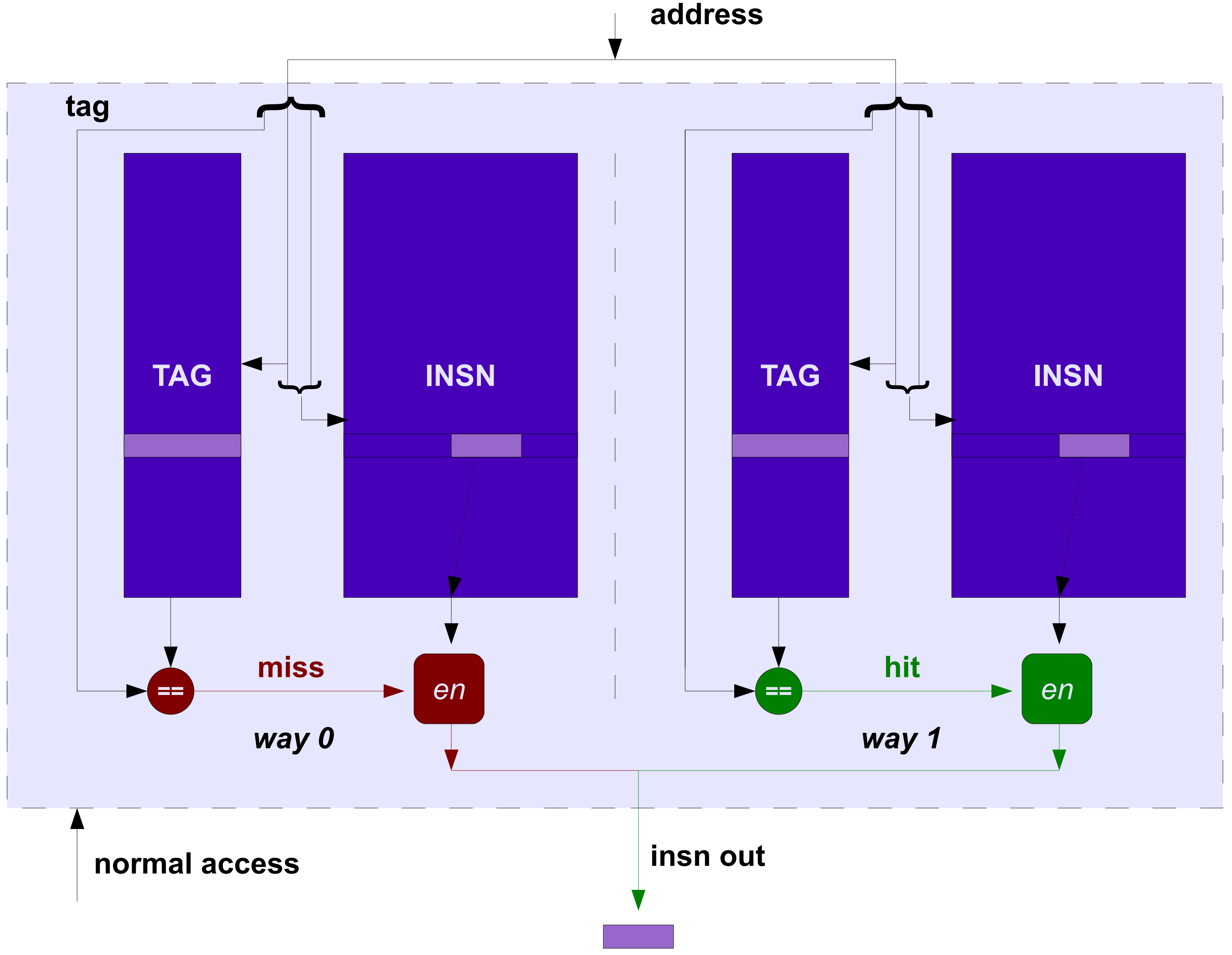
All Tag and Insn Banks Accessed



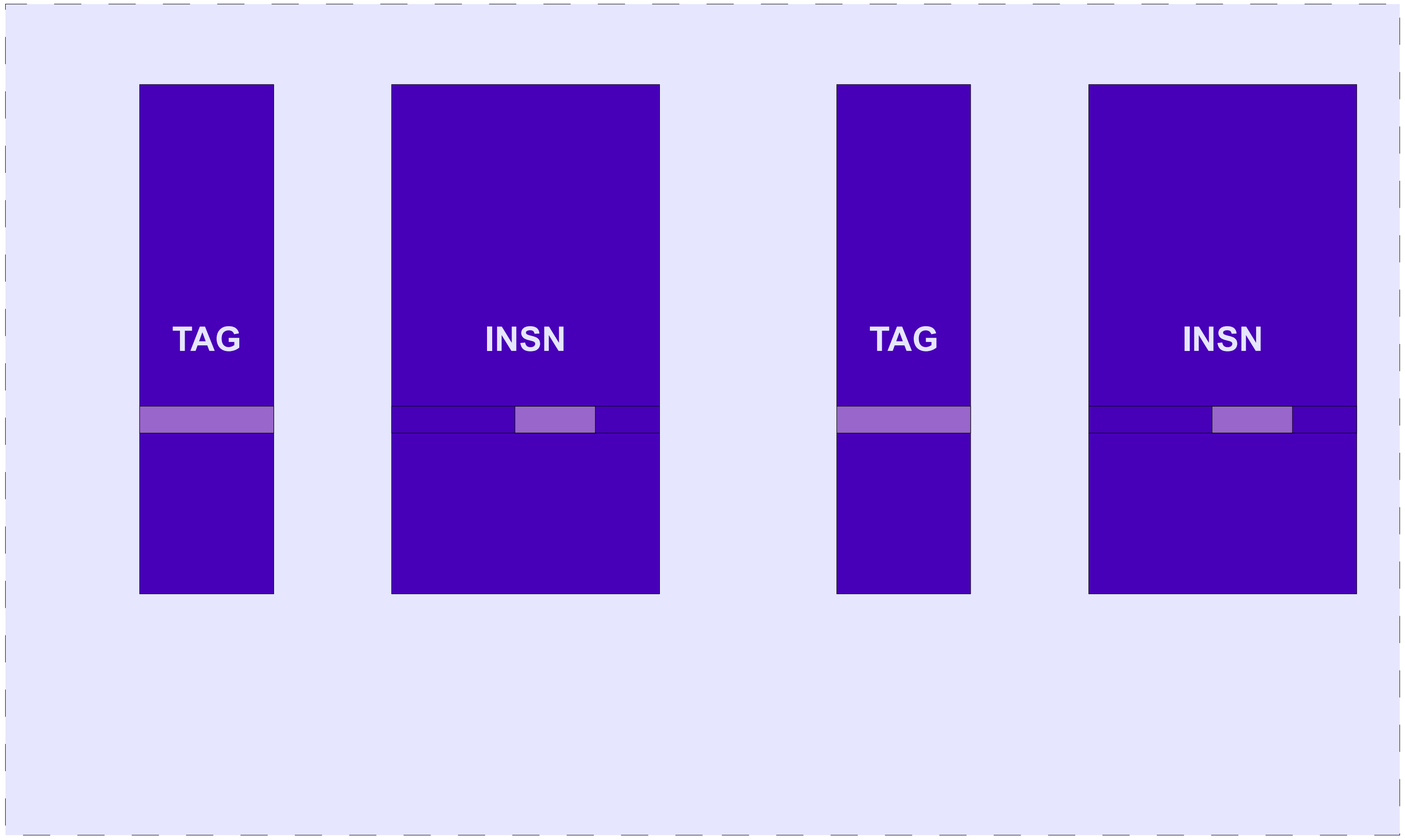
Tag Checks Performed



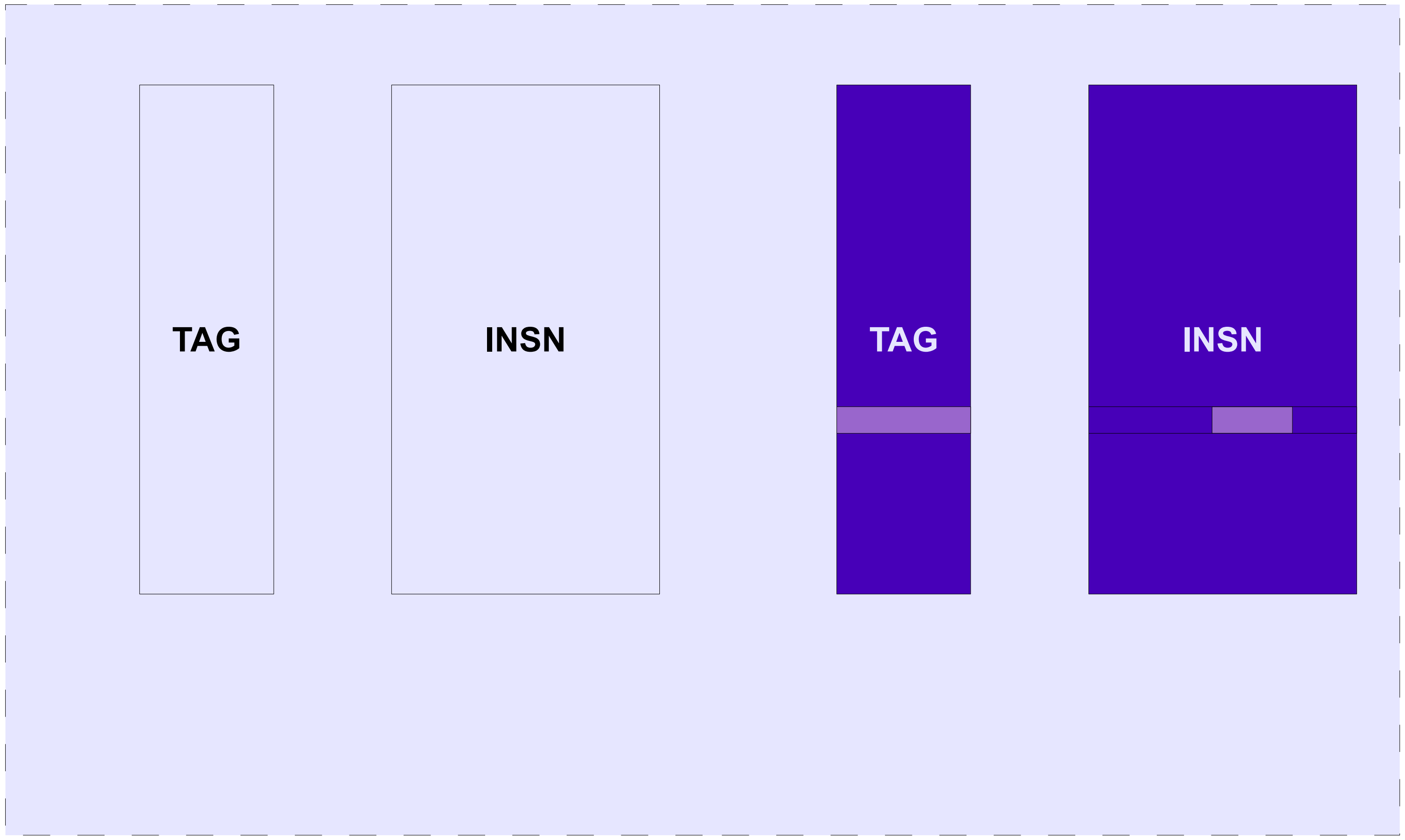
Instruction Read Out



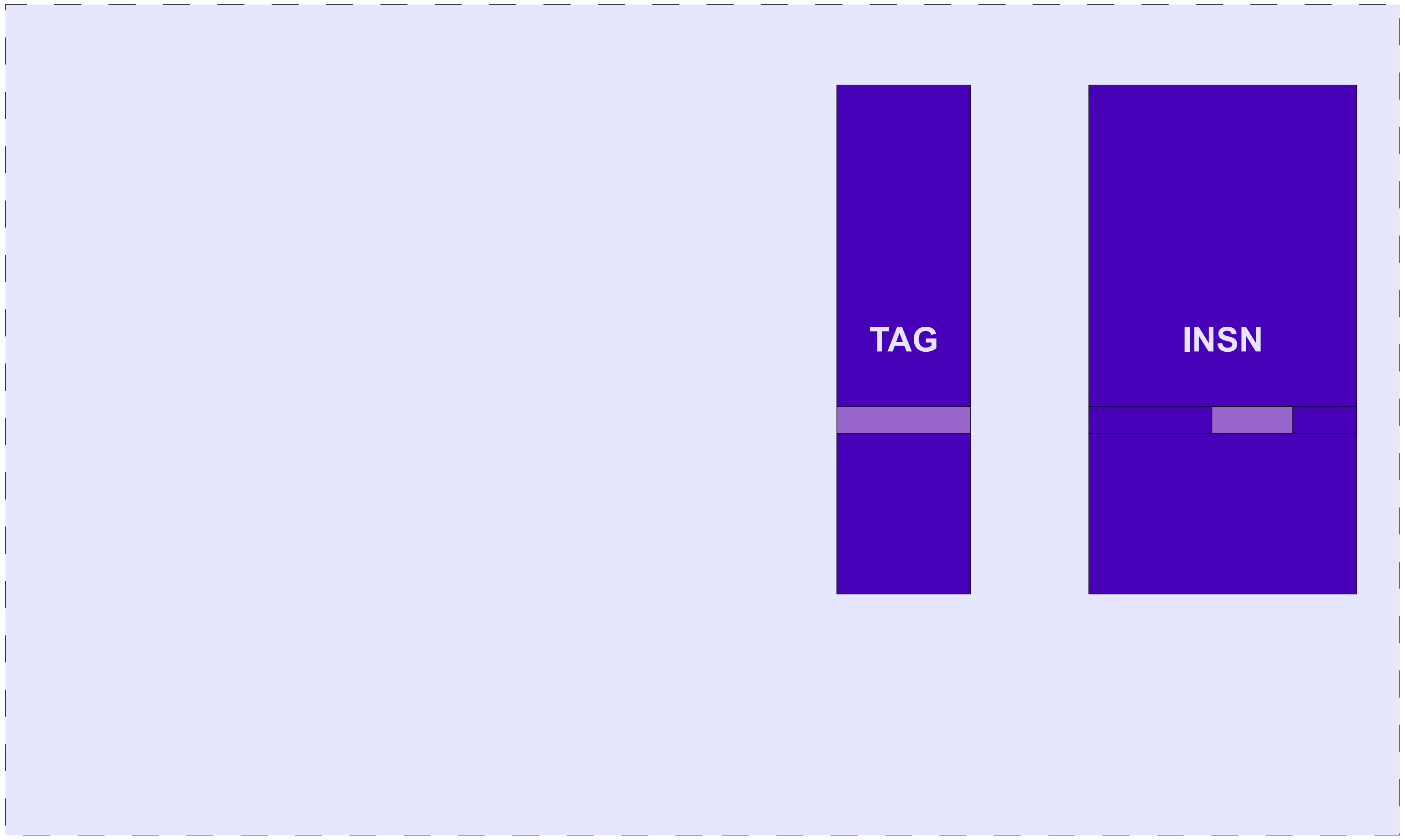
Currently Wasteful If Instruction Location is Known



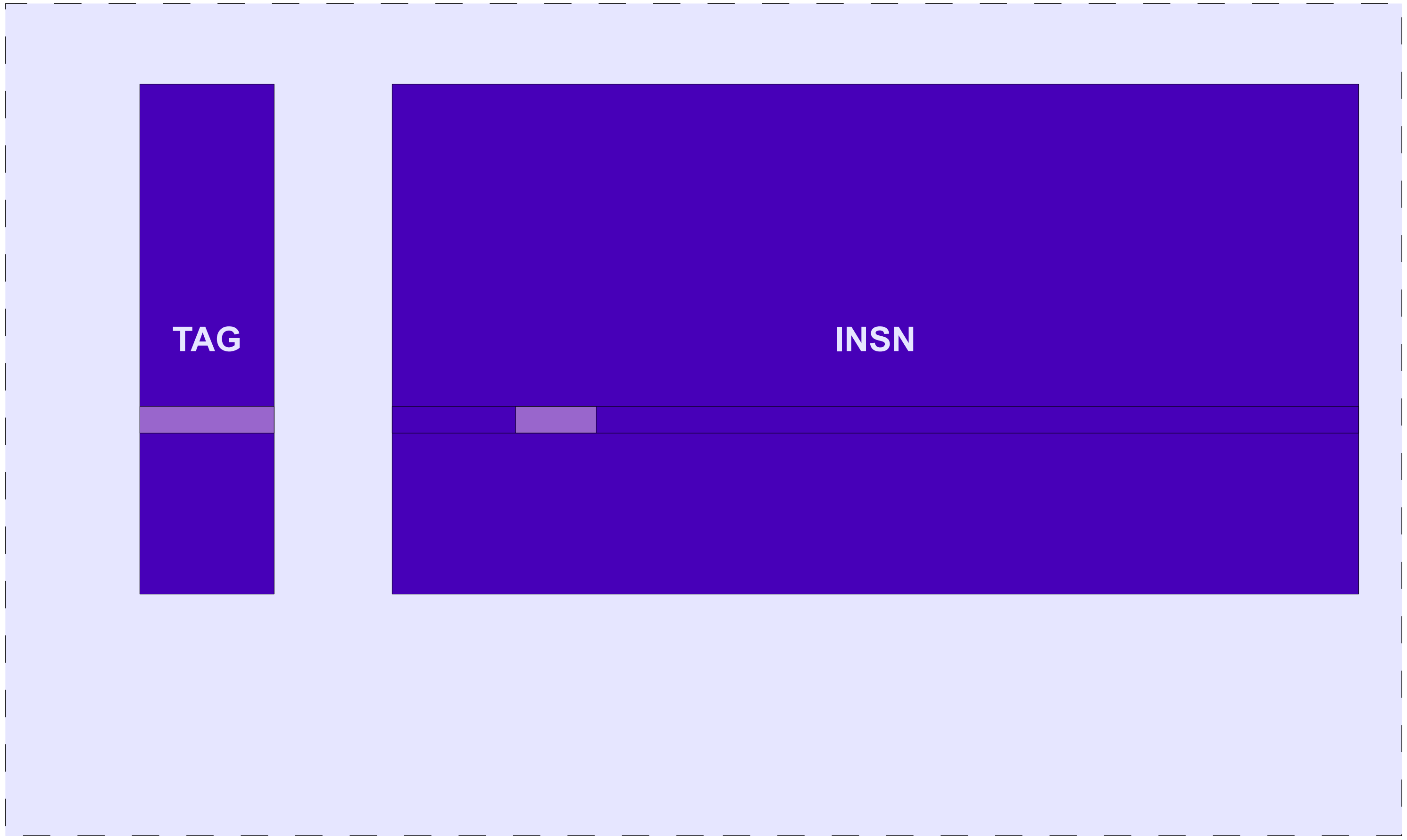
We Can Do Better: Way Prediction



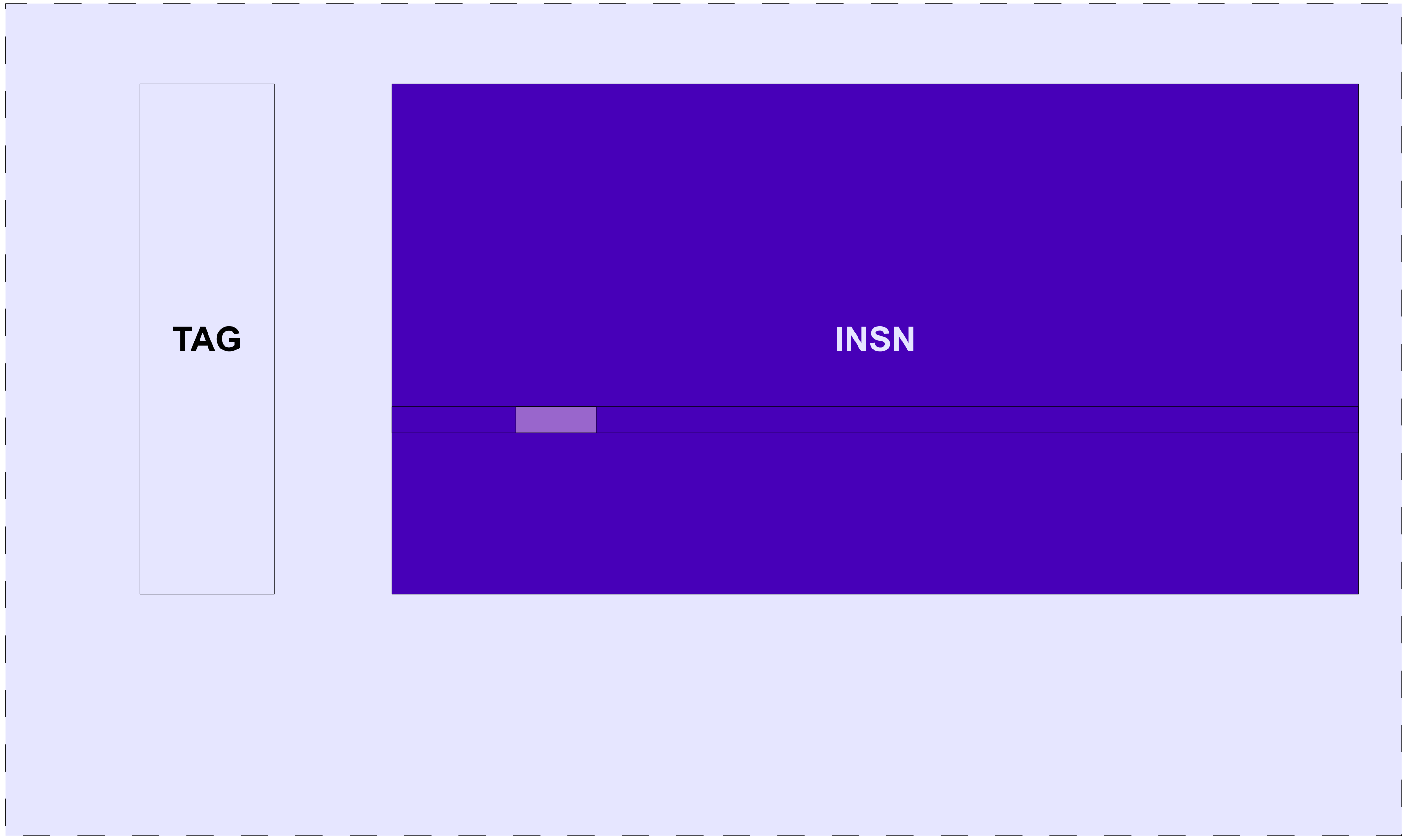
We Can Do Better: Way Prediction



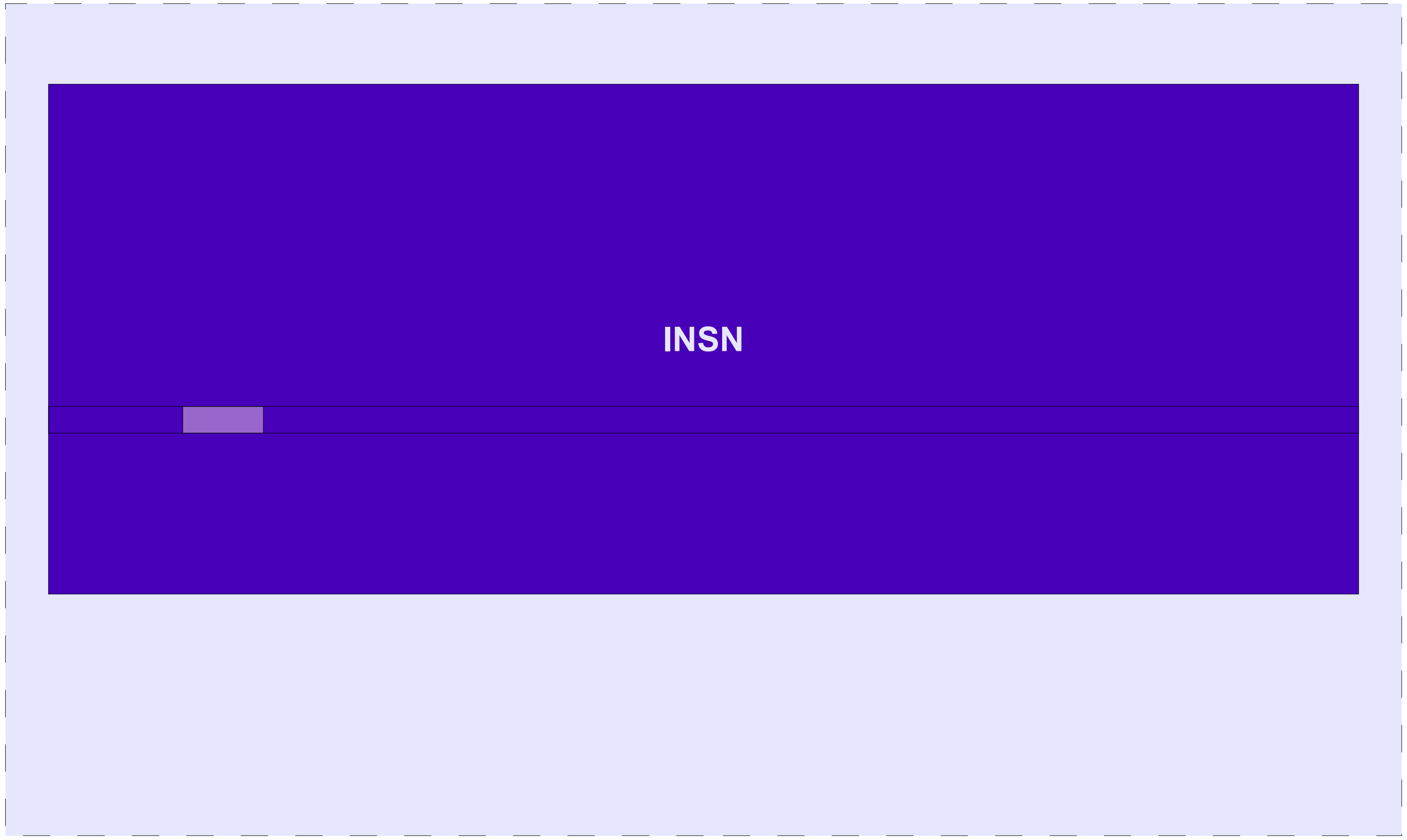
We Can Do Better: Way Prediction



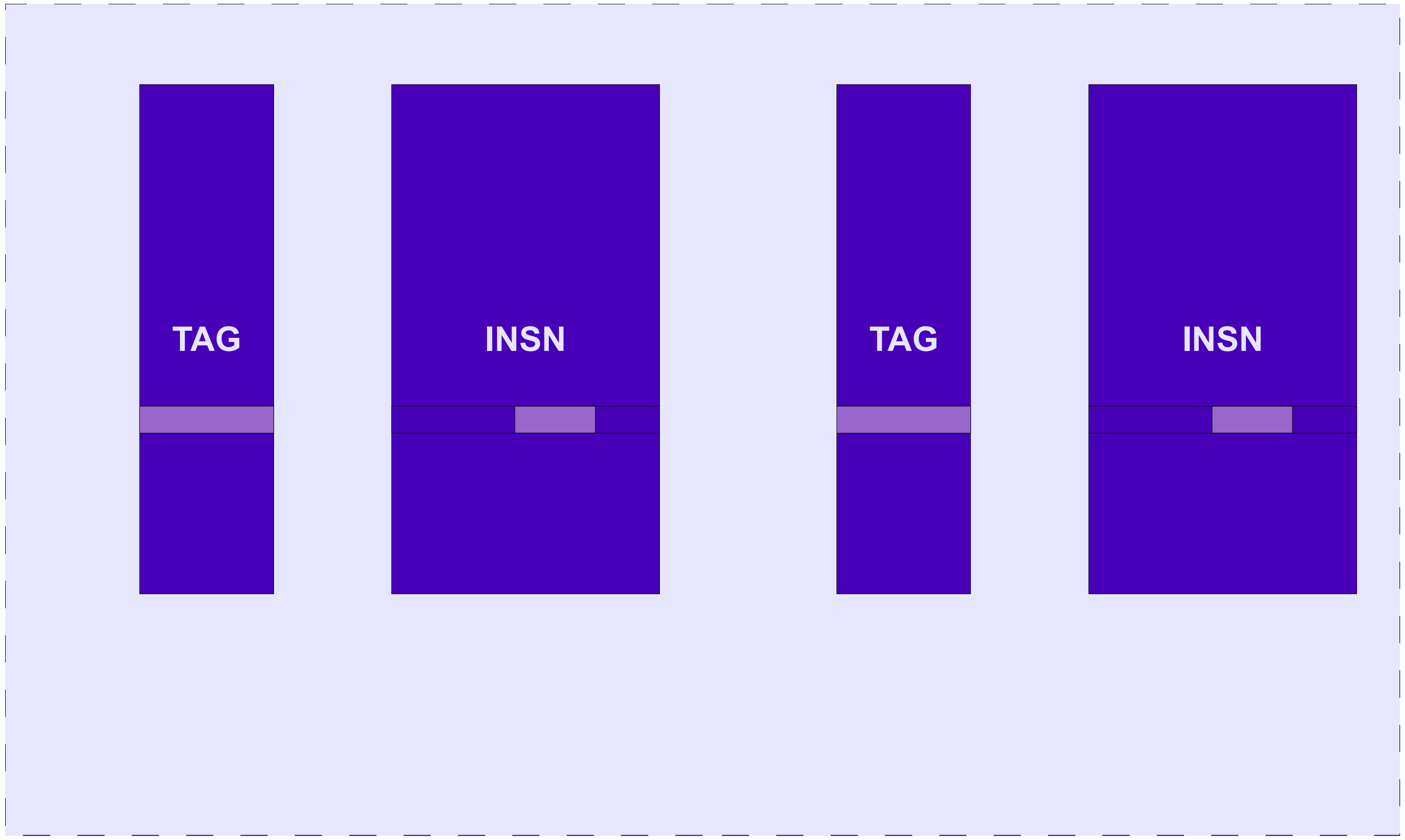
Even Better: Tagless Accessing



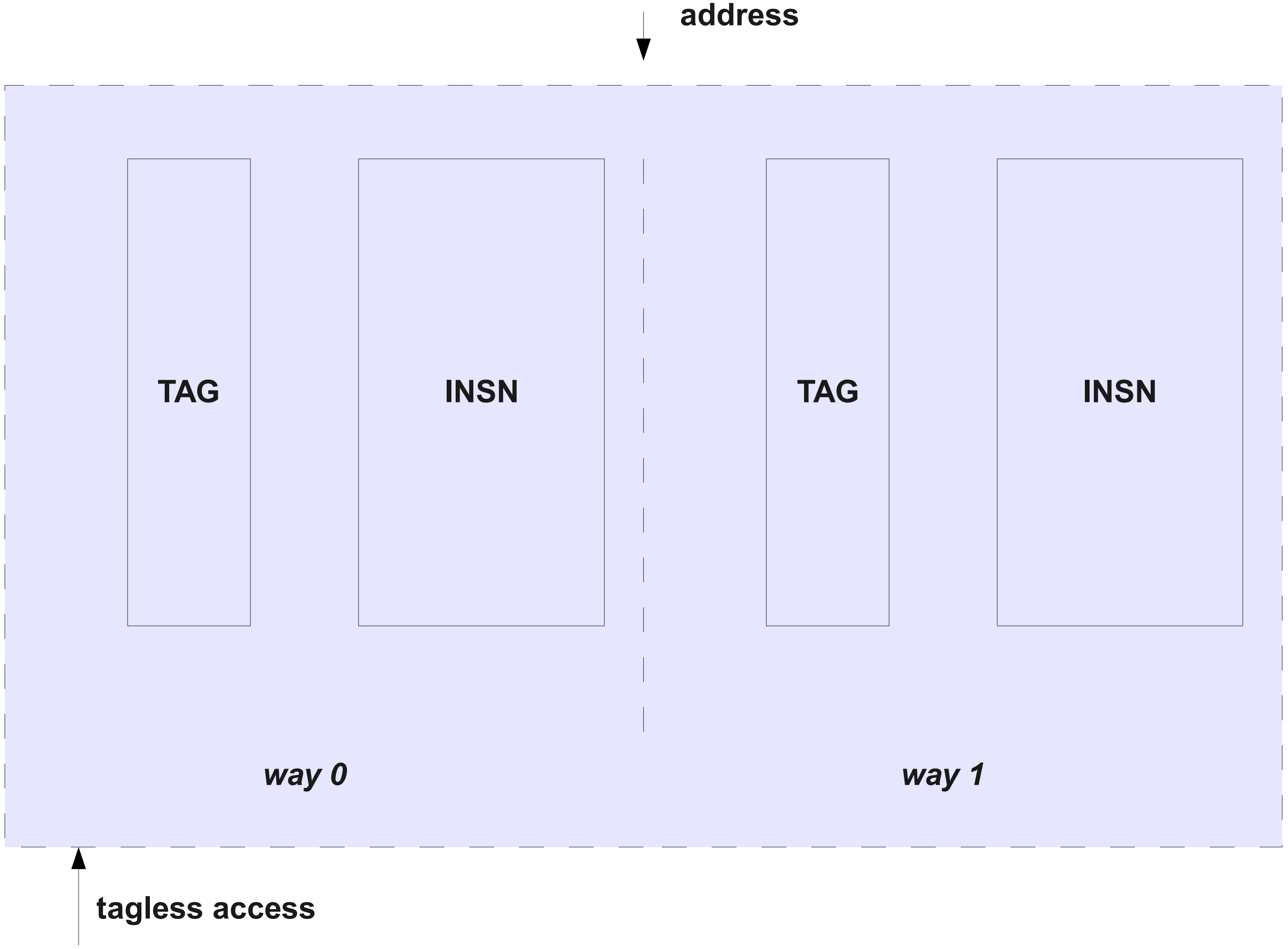
Cache Behaves Like A Scratchpad



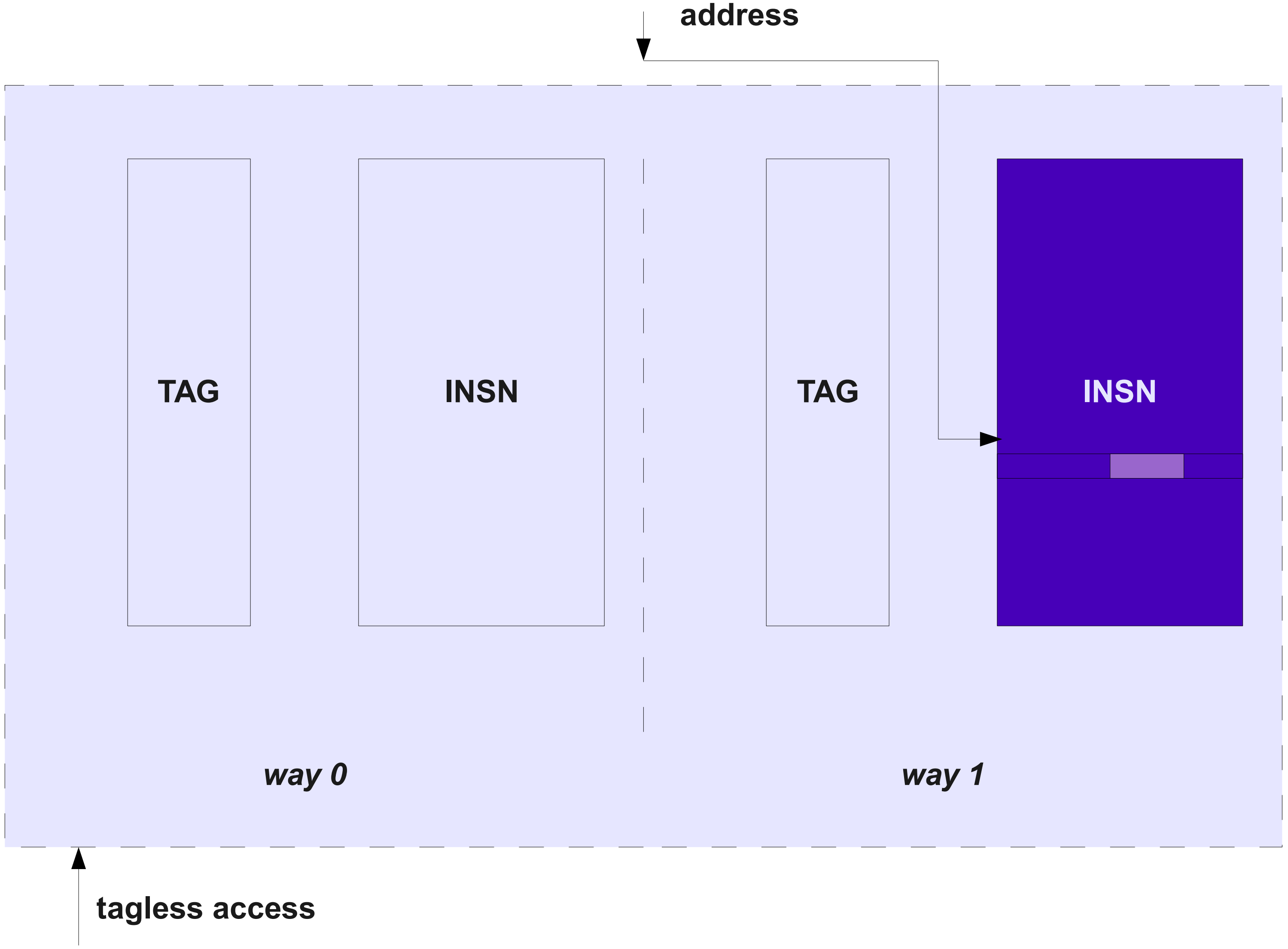
Best of Both Worlds



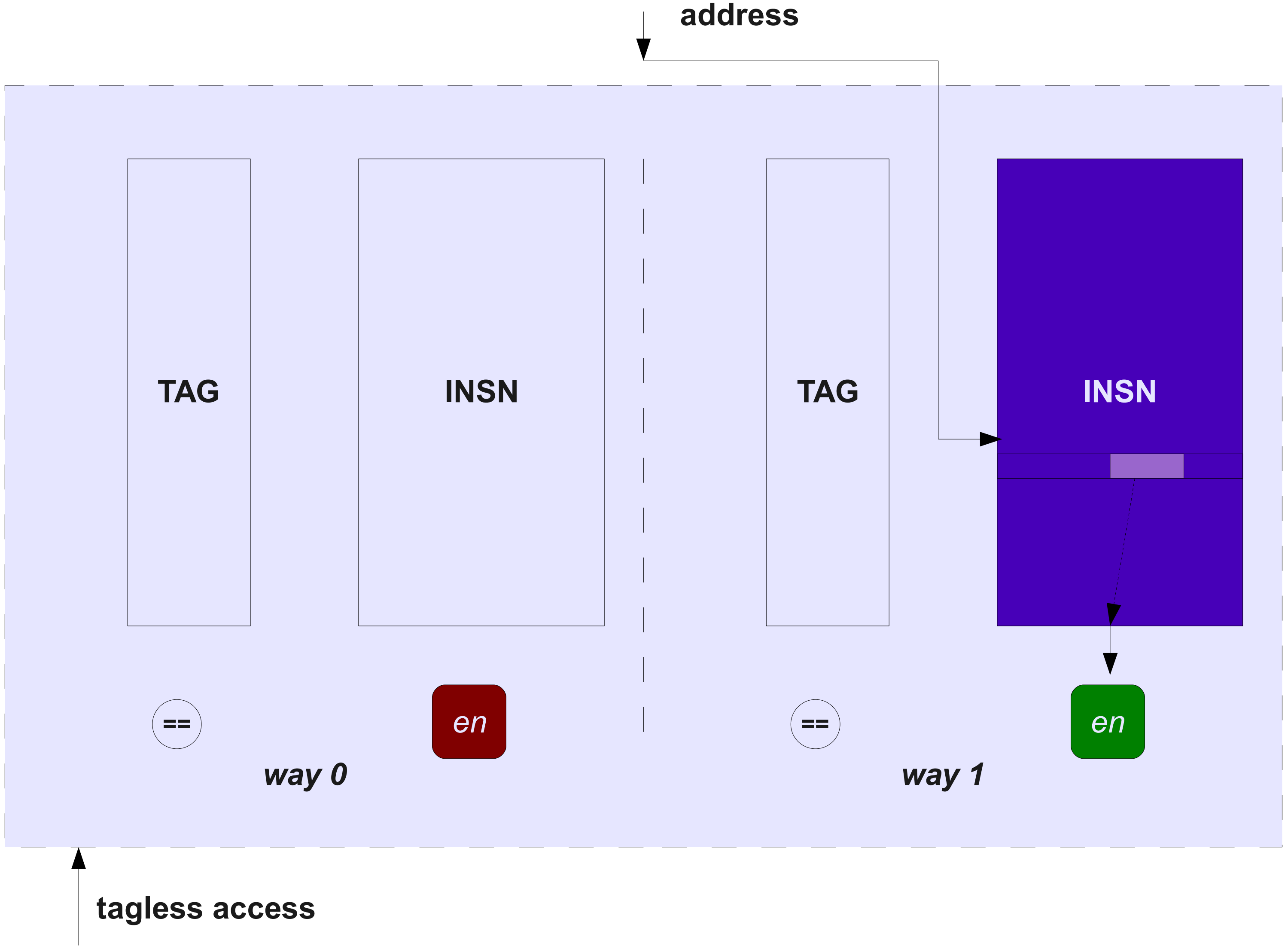
Tagless Access



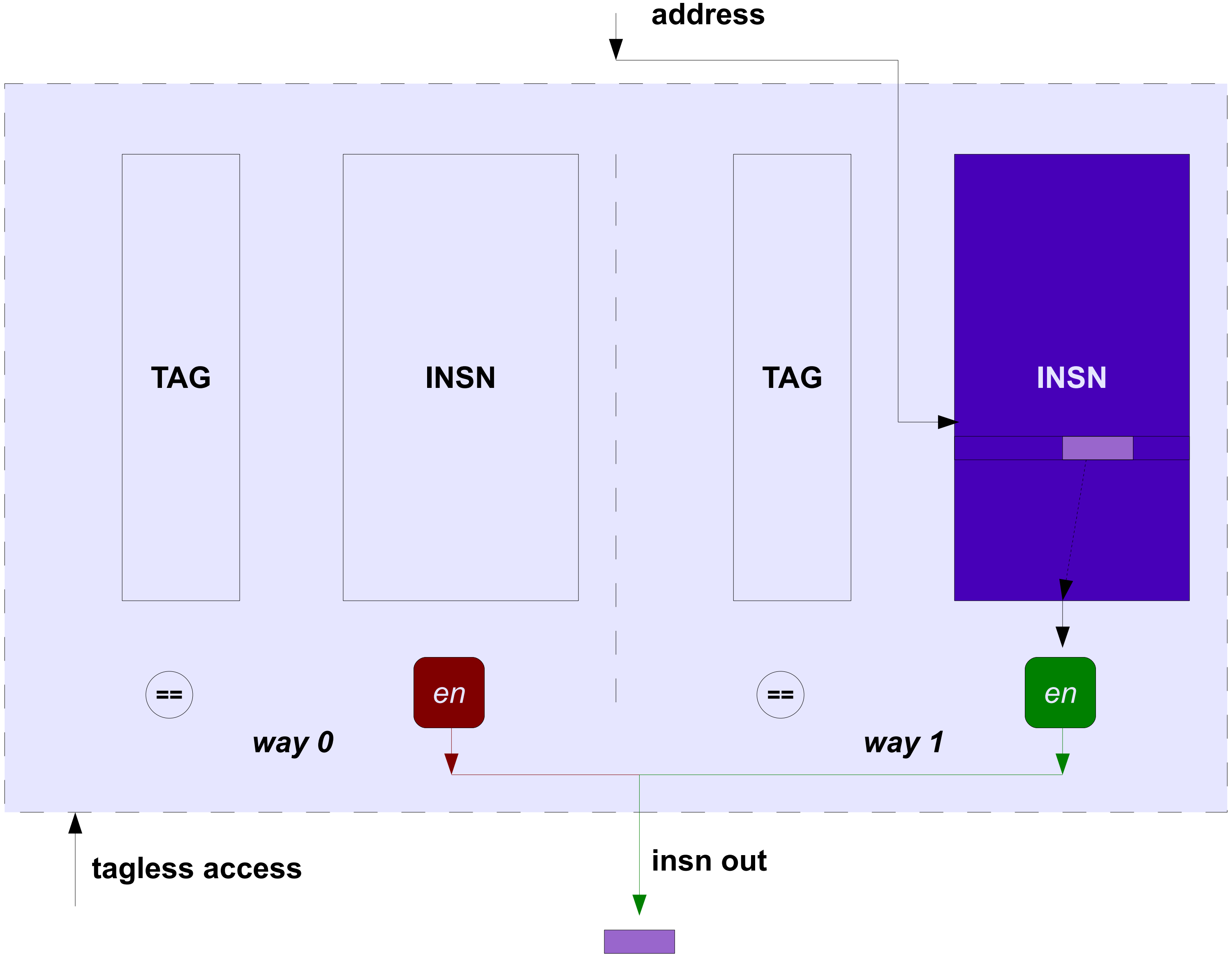
Only One Bank Accessed



No Tag Checks Required



Instruction Read Out



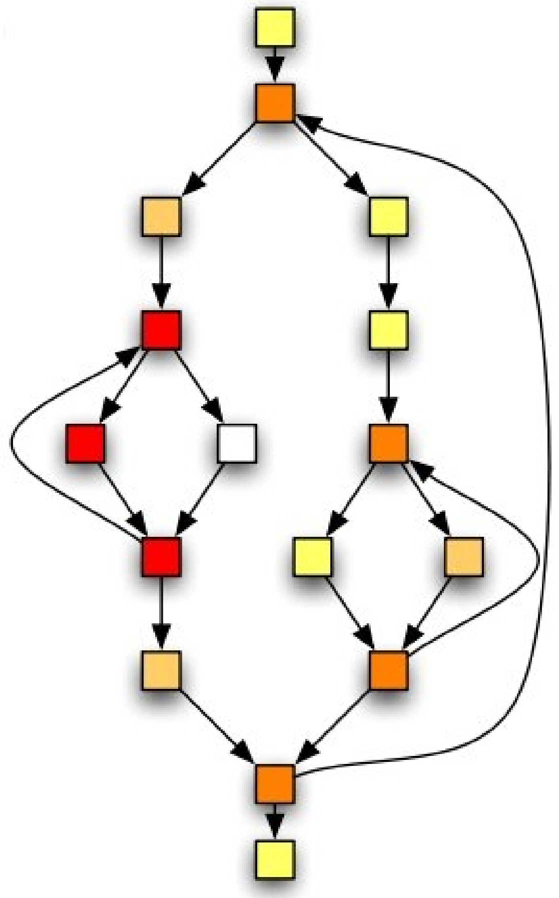
Tagless Accessing

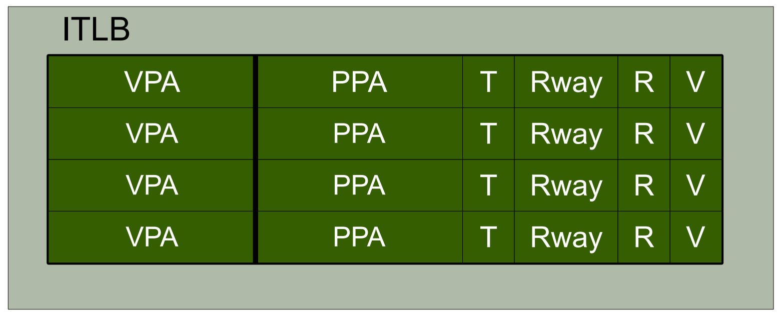
- **Use cache like a scratchpad when possible**
  - For power saving
- **Use like a normal cache at other times**
  - When flexibility is required
- **Requires careful management**
  - Performance losses otherwise

Outline

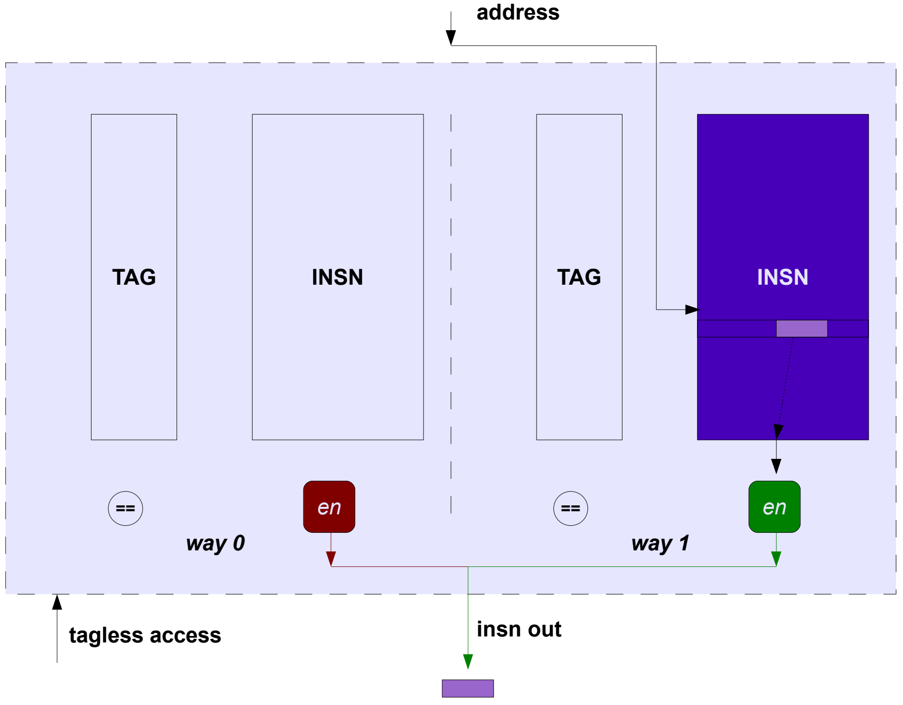
- Instruction cache power usage
- **Tagless instruction caching**
  - Link-time optimisation
  - Hardware modifications
- Evaluation
- Conclusions

Two Aspects



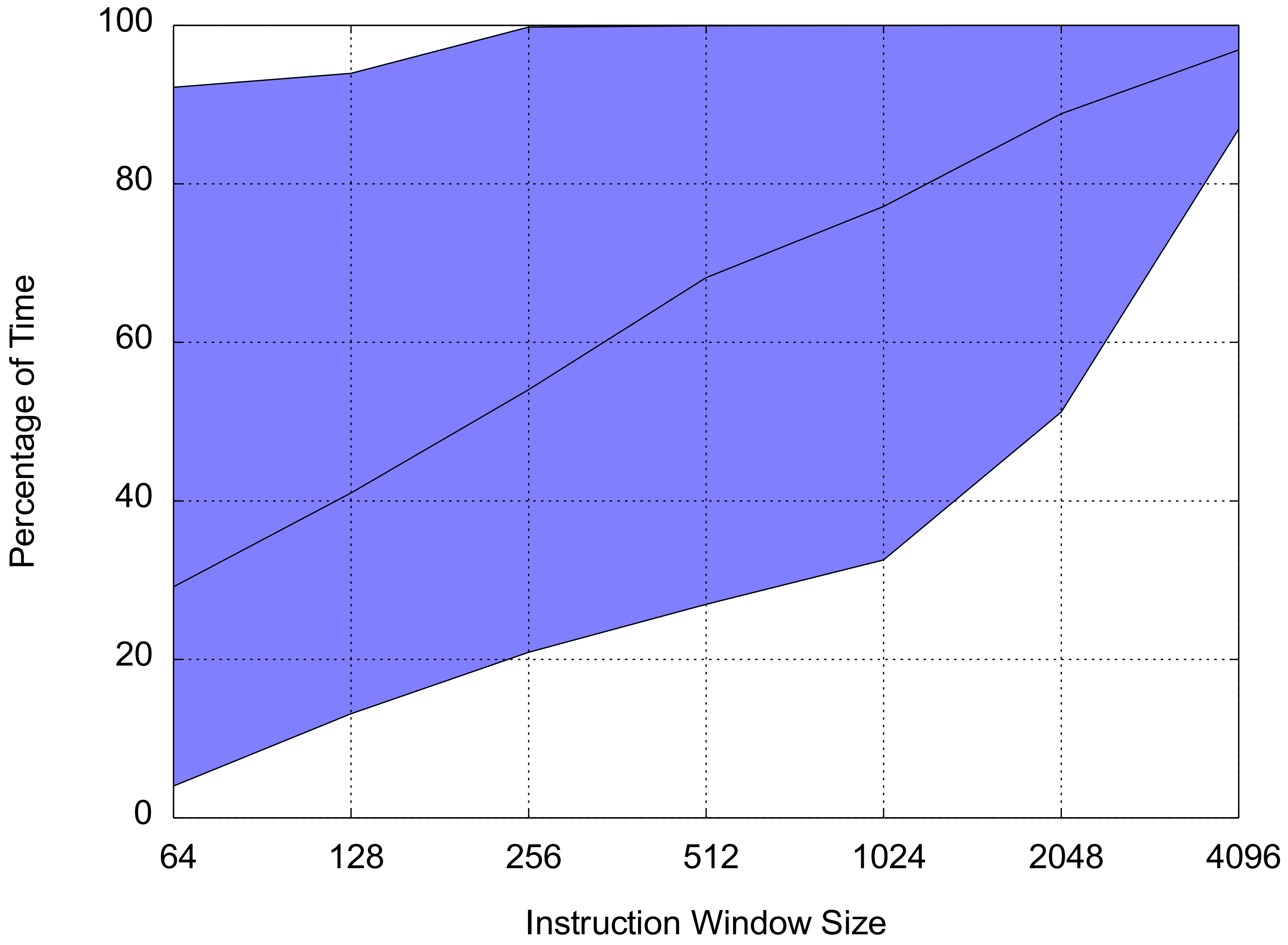


VPA	PPA	T	Rway	R	V
VPA	PPA	T	Rway	R	V
VPA	PPA	T	Rway	R	V
VPA	PPA	T	Rway	R	V



UNIVERSITY OF CAMBRIDGE

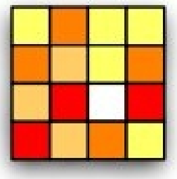
Where Can The Compiler Help?



Compiler Algorithm

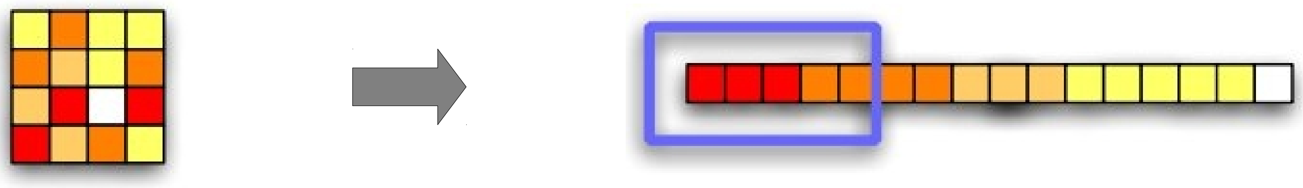
- **Our scheme uses new code placement algorithm**
  - Converts temporal locality to spatial locality
  - Group frequently executed code into regions
    - Access without tag checks
    - Others accessed normally (with tag checks)
- **Algorithm goals**
  - Spend as much time as possible in tagless regions
  - Minimise switching between regions
  - Fill regions as much as possible

Example - Algorithm



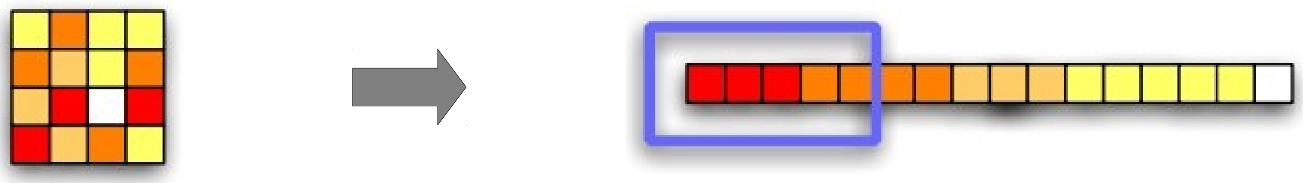
Original binary used as input

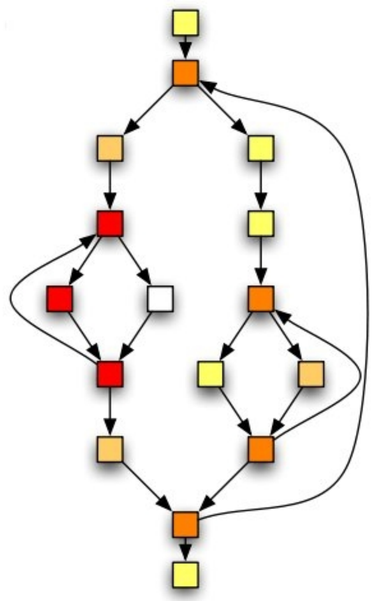
Example - Algorithm



Blocks sorted by execution frequency

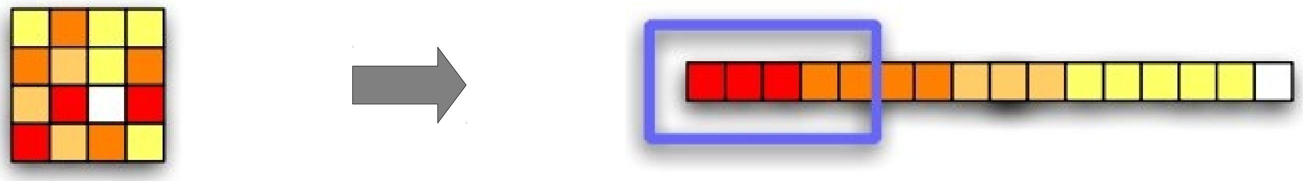
Example - Algorithm



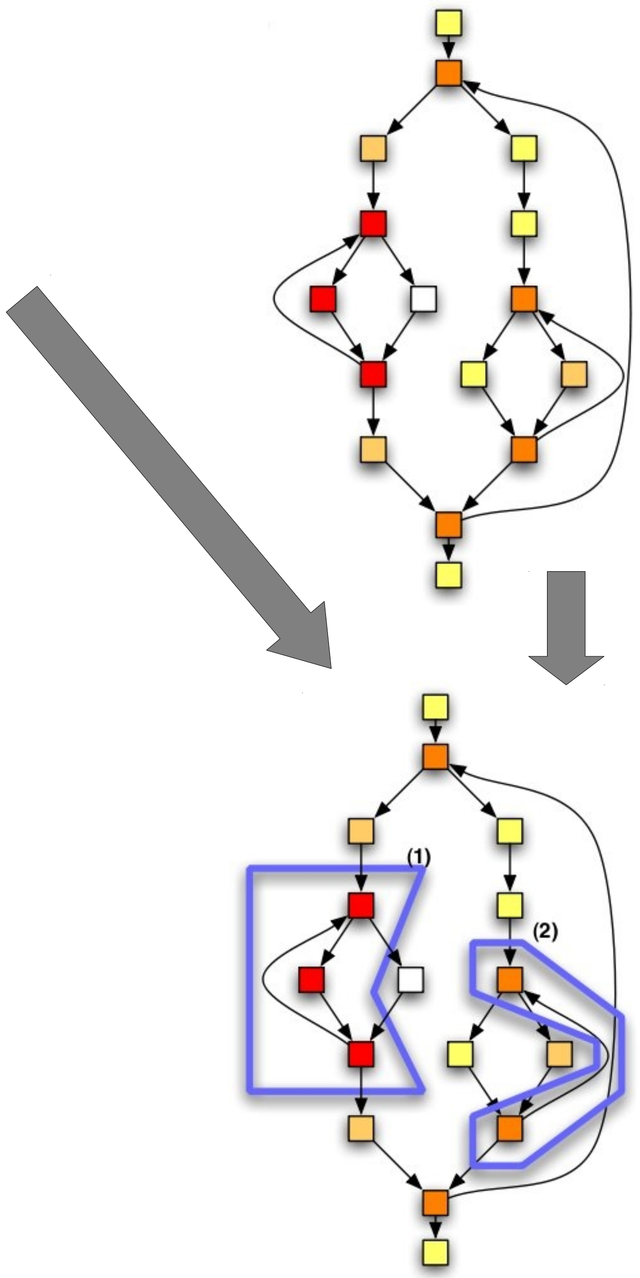


Control flow graph constructed

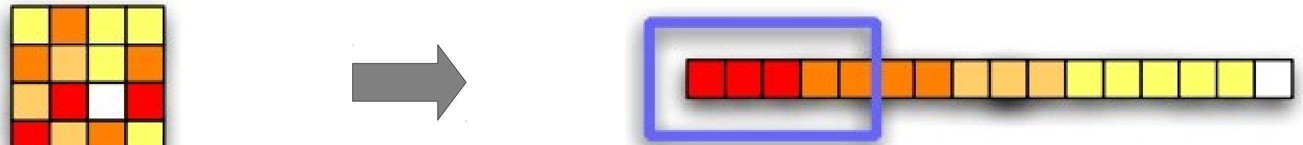
Example - Algorithm



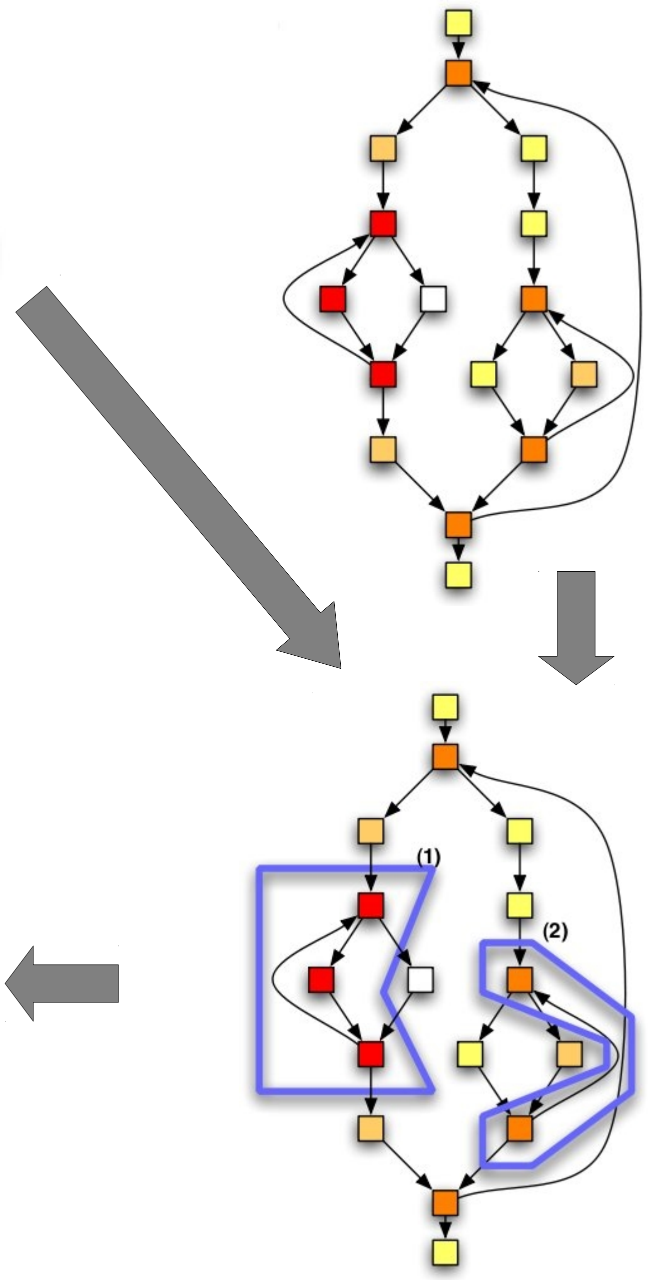
Initial clusters created

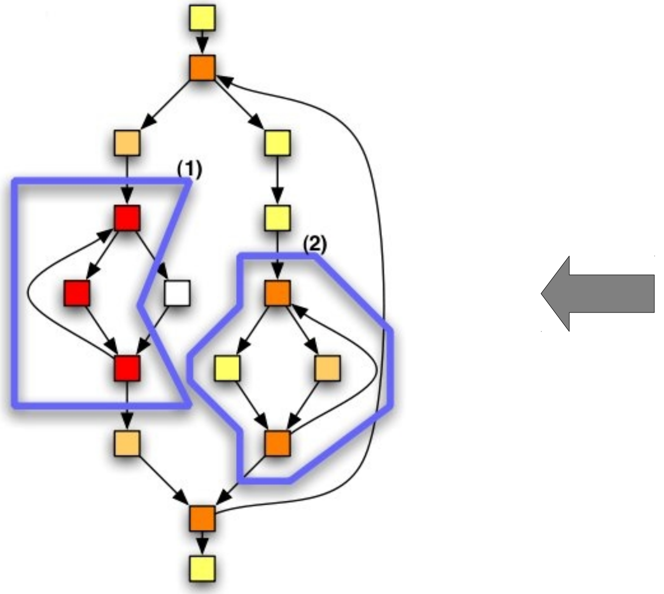


Example - Algorithm

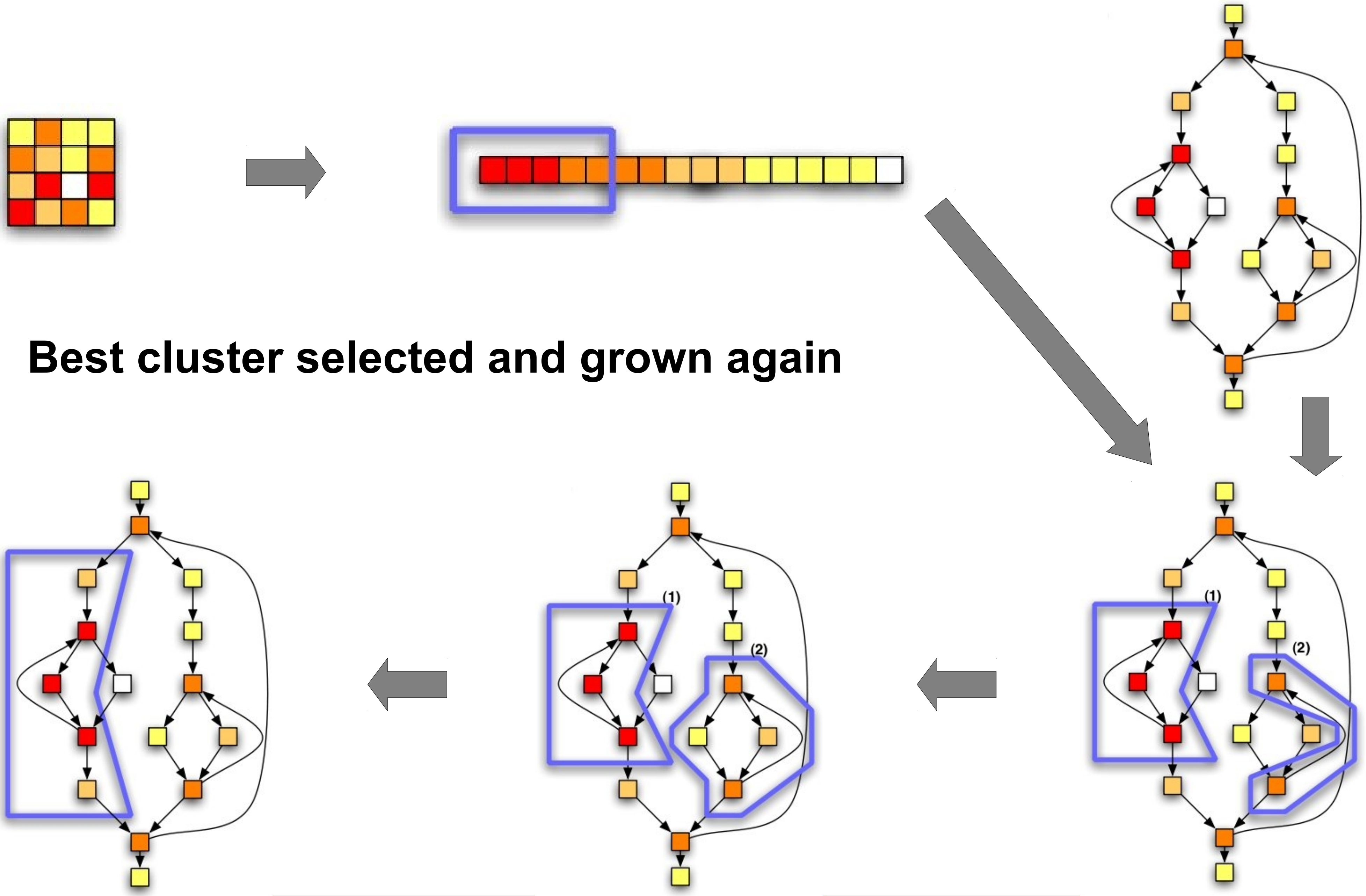


Clusters expanded where profitable





Example - Algorithm

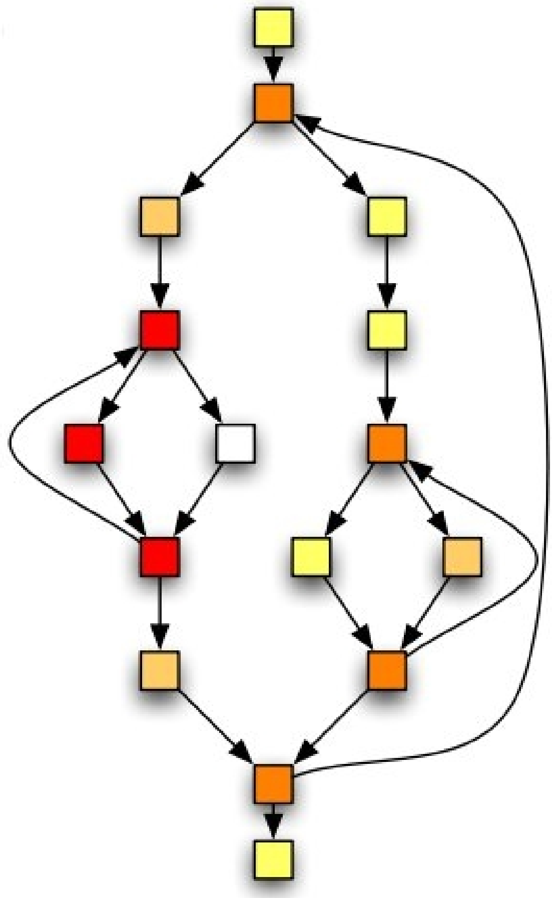


Best cluster selected and grown again

UNIVERSITY OF CAMBRIDGE

Link-Time Optimisation Summary

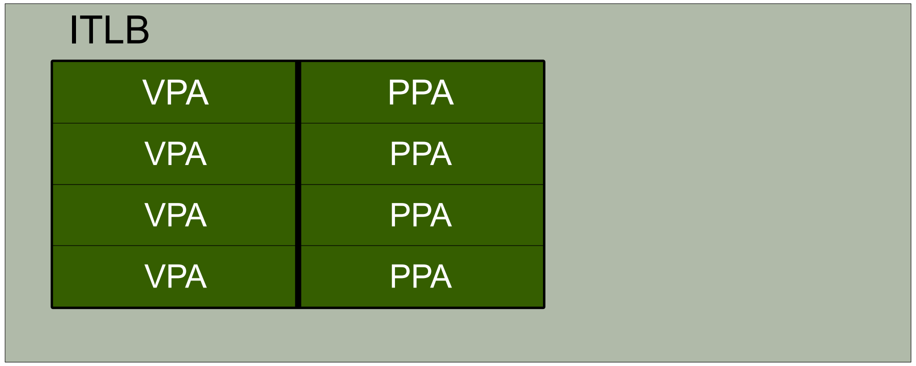
- **Spatial locality from temporal**
- **Hot basic blocks in regions**
  - No tag checks
- **Maximise region code**
- **Minimise region switching**

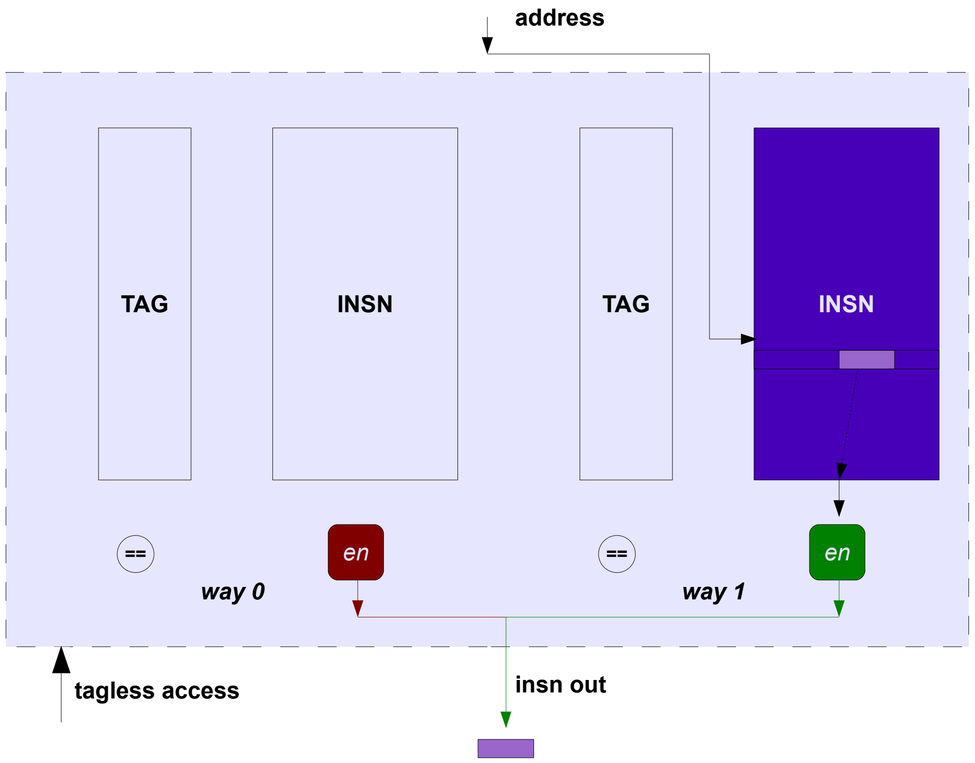


Additional Hardware Support

- **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

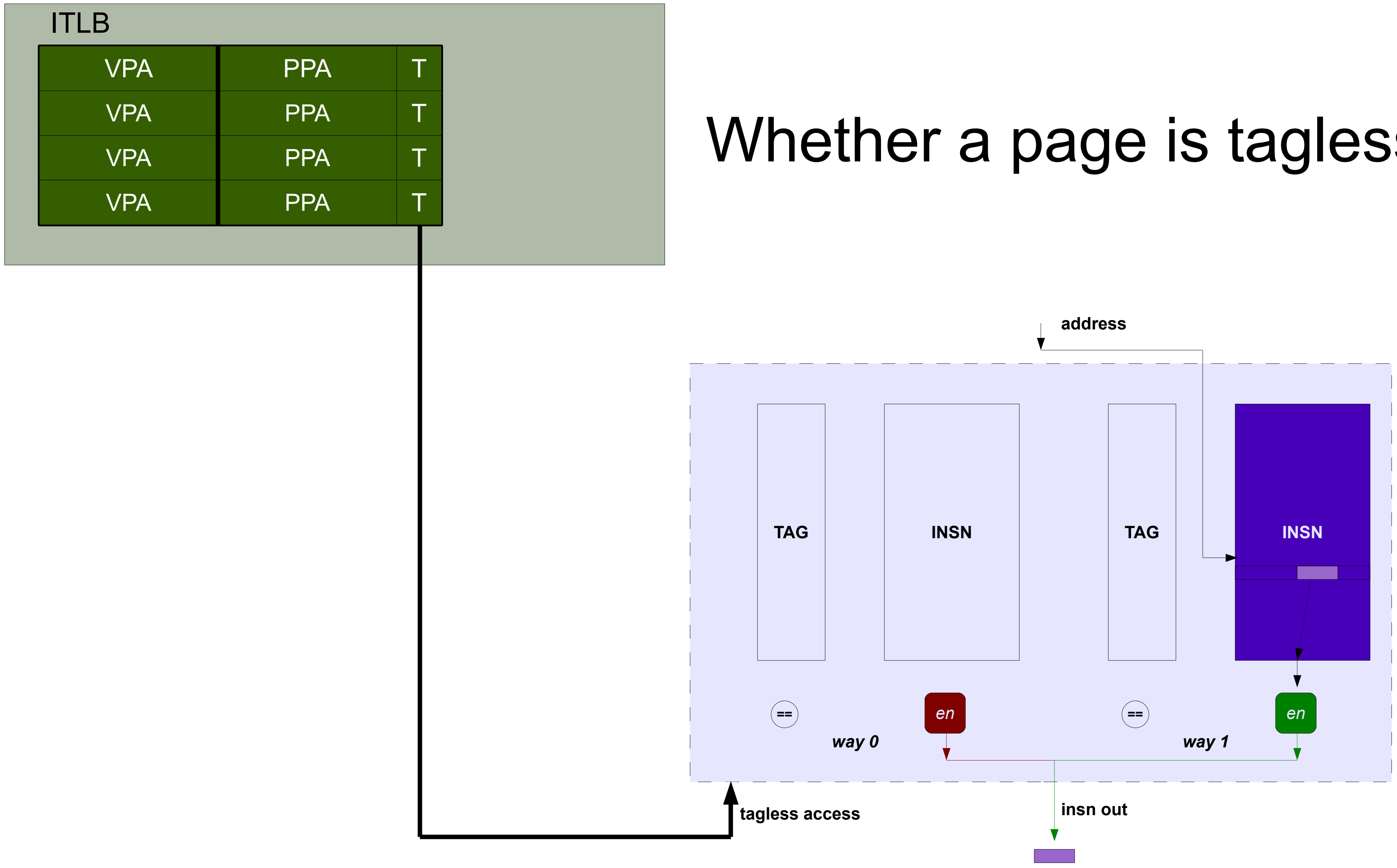
ITLB Support





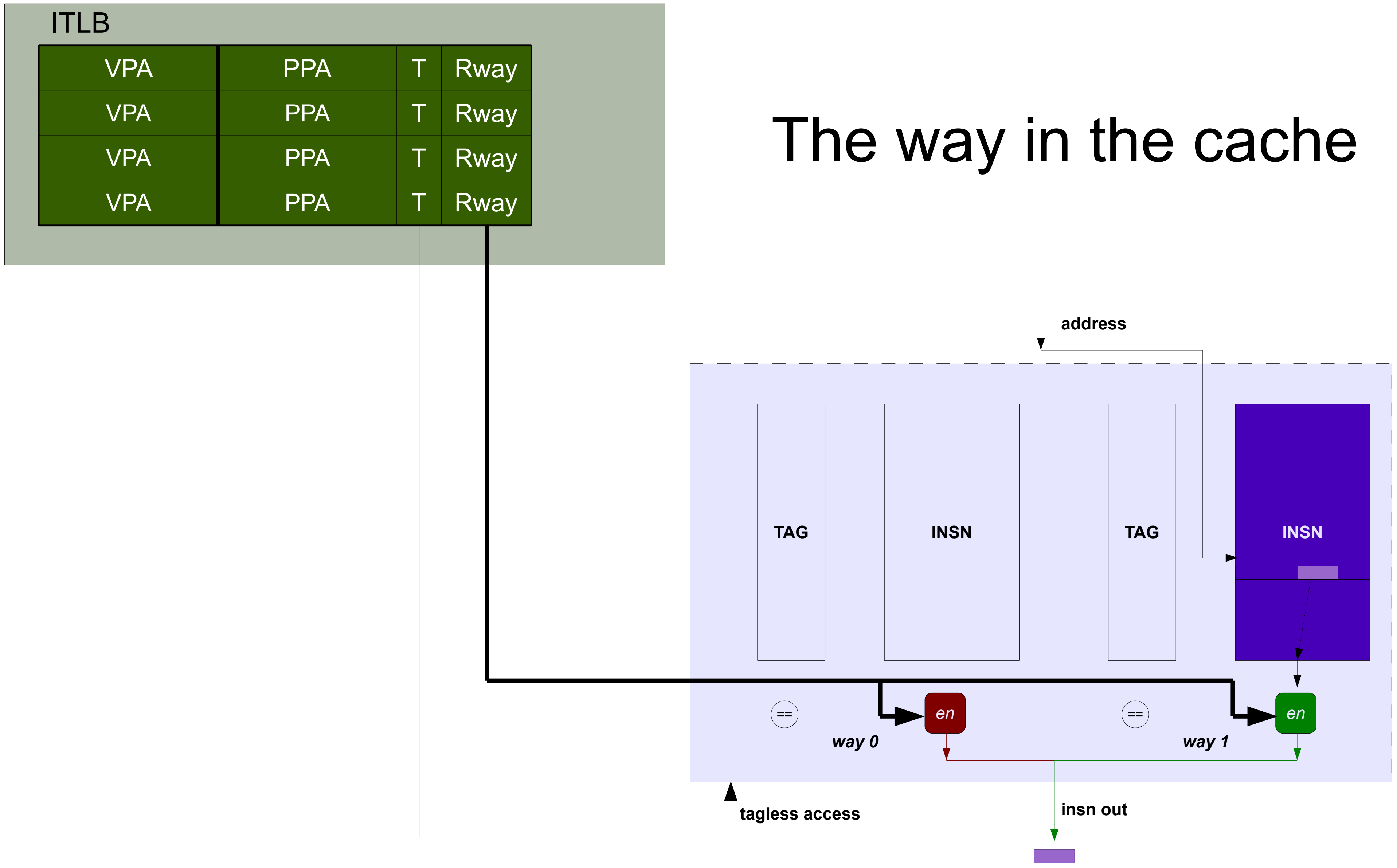
ITLB Support

Whether a page is tagless



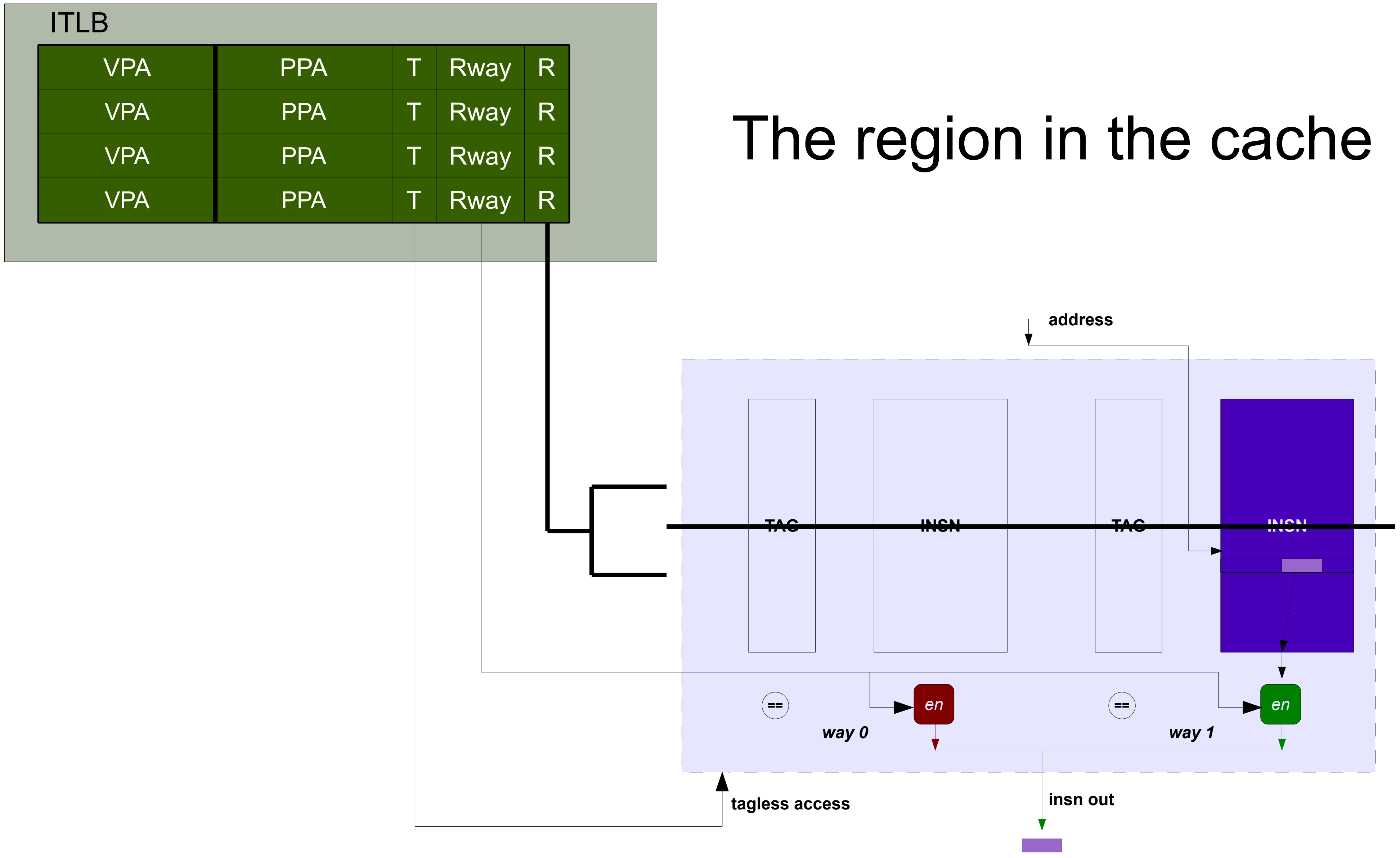
UNIVERSITY OF CAMBRIDGE

ITLB Support



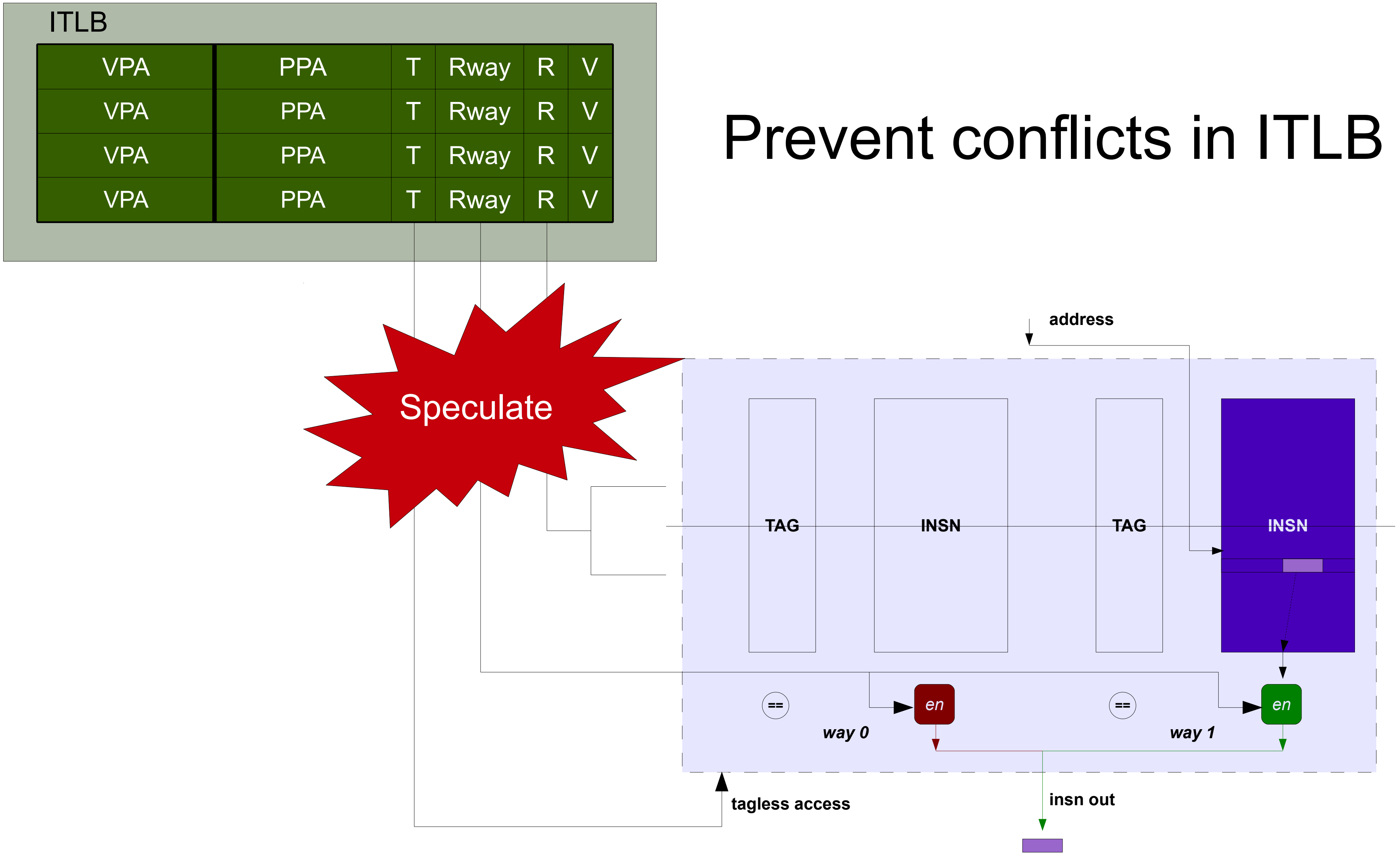
The way in the cache

ITLB Support



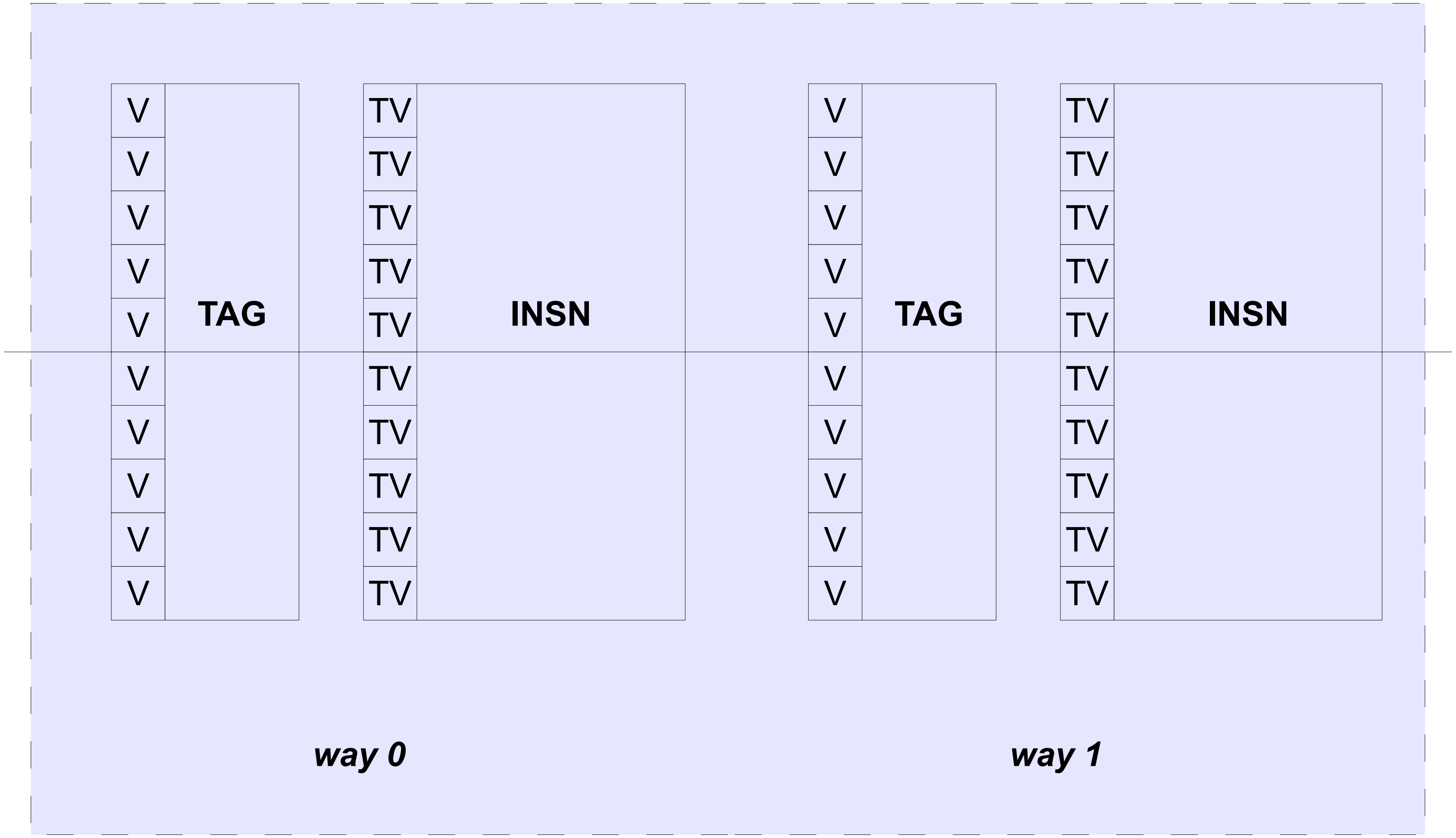
The region in the cache

ITLB Support



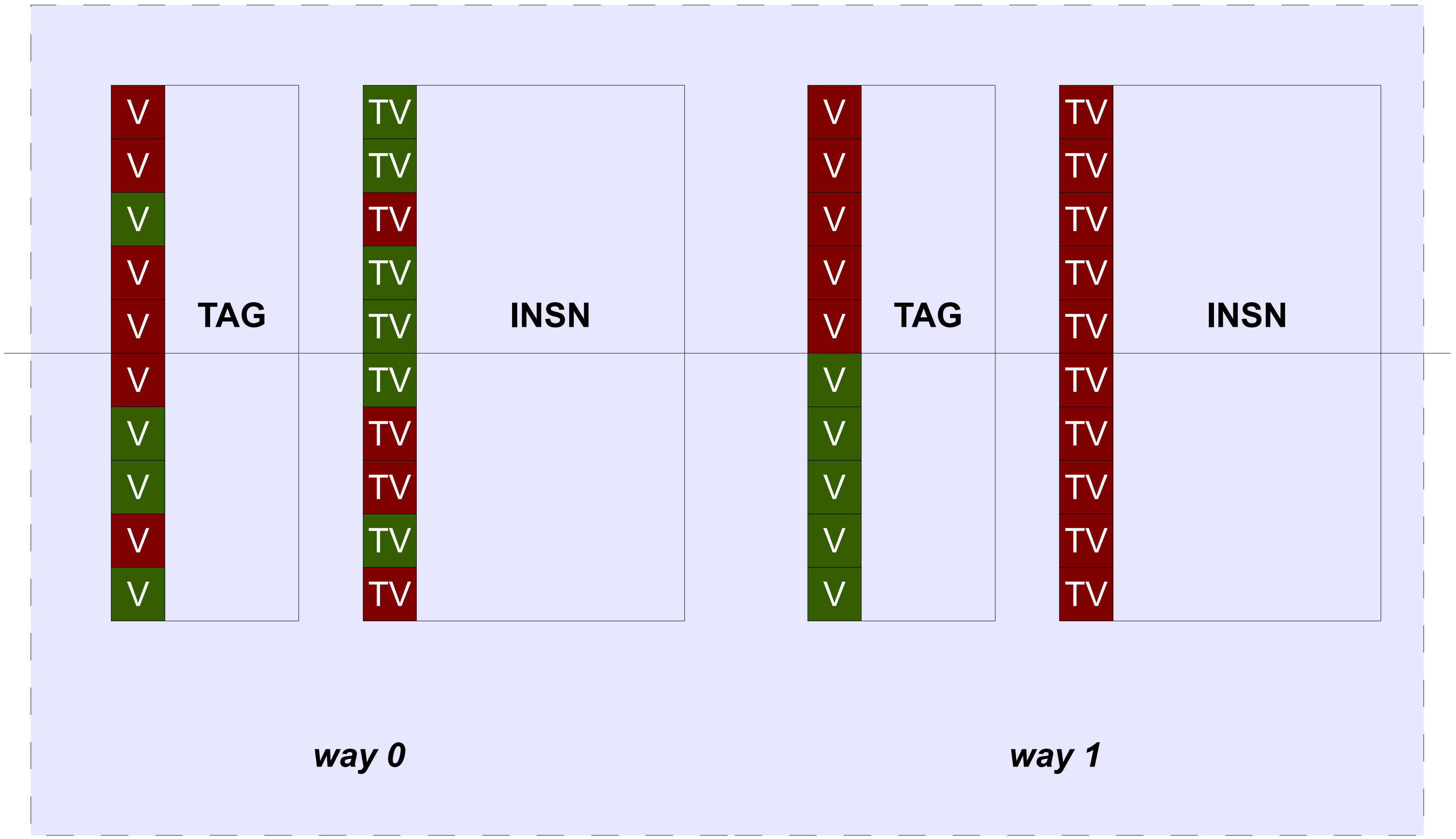
Prevent conflicts in ITLB

Instruction Cache Support



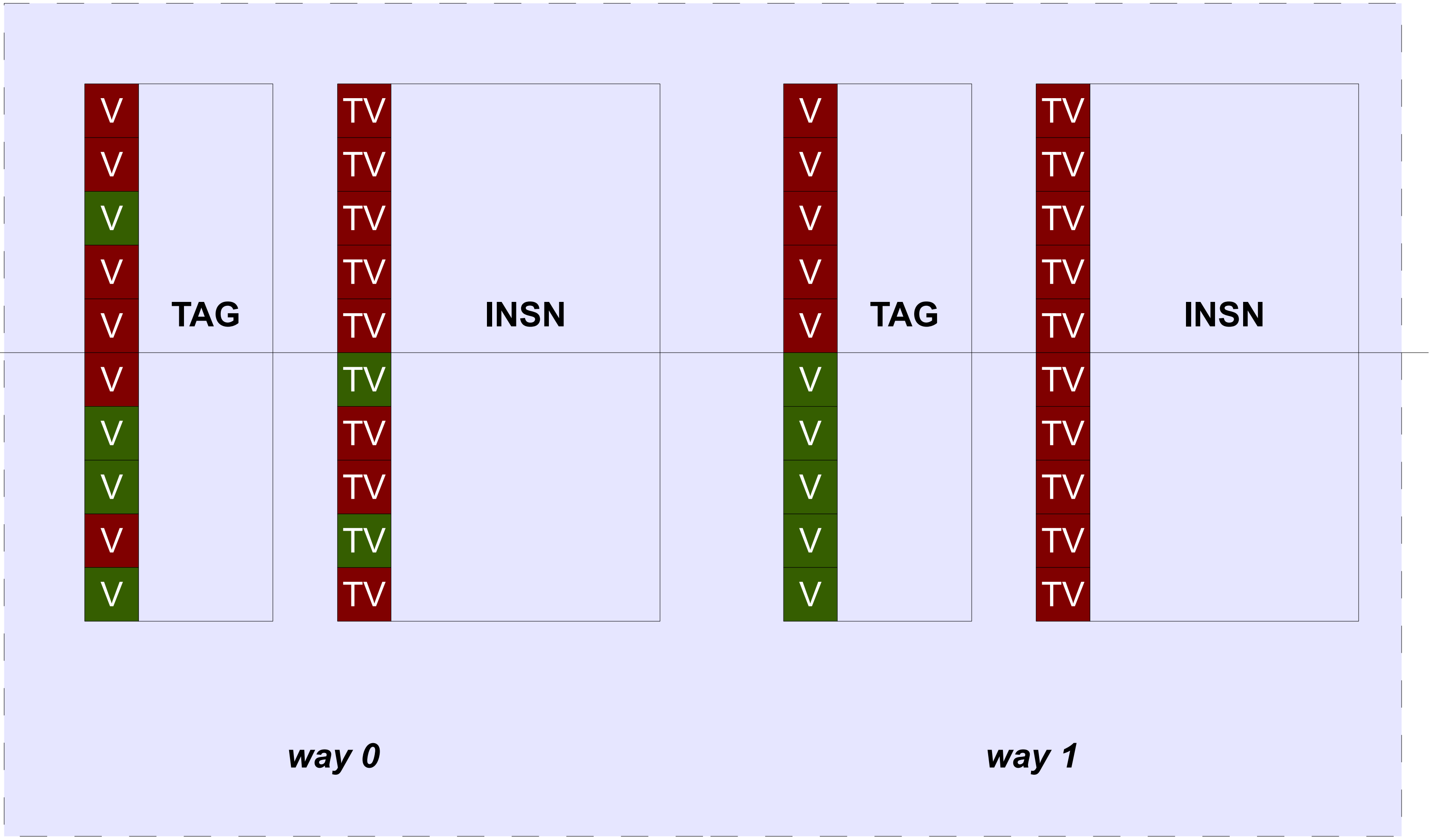
Tagless valid bits mark lines as valid in tagless mode

Instruction Cache Support



Tagless and tagged instructions can coexist

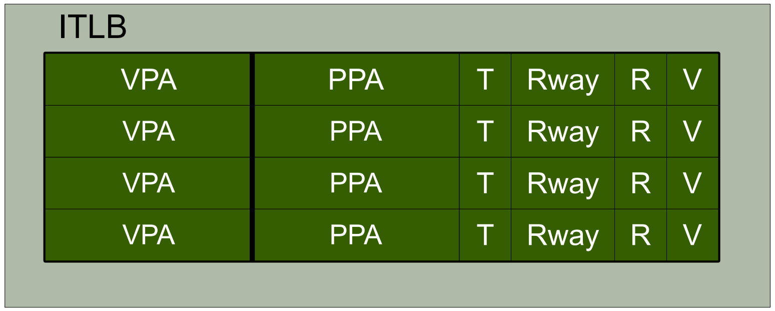
Instruction Cache Support



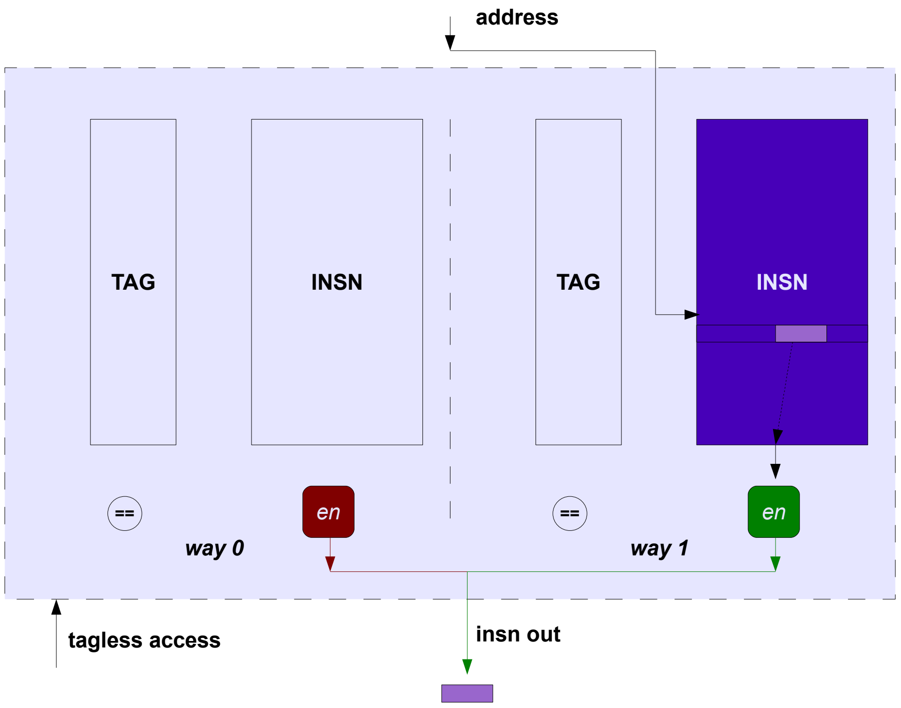
Conflicts reset the region's tv-bits

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**


ITLB

VPA	PPA	T	Rway	R	V
VPA	PPA	T	Rway	R	V
VPA	PPA	T	Rway	R	V
VPA	PPA	T	Rway	R	V



UNIVERSITY OF CAMBRIDGE

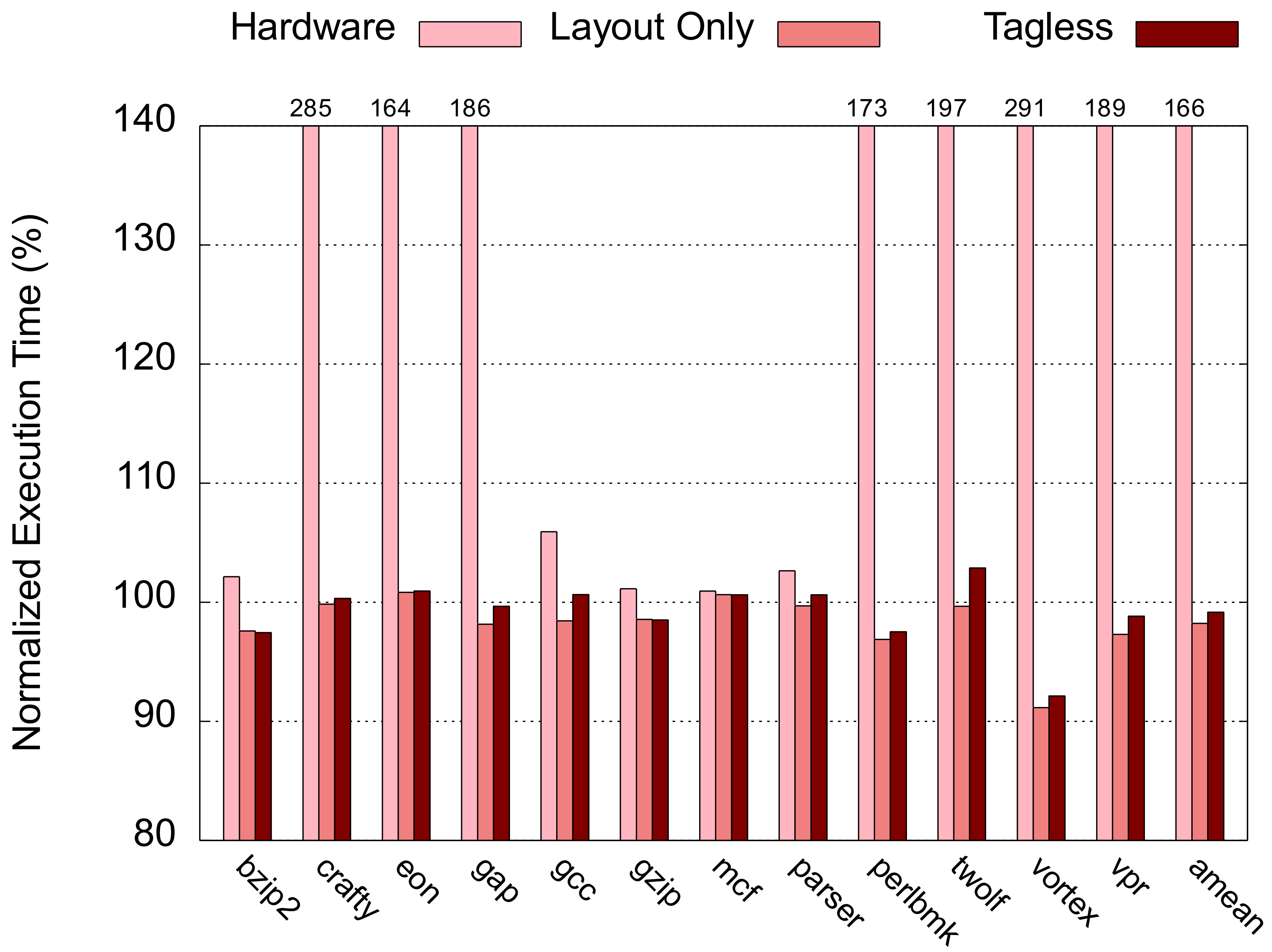
Outline

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

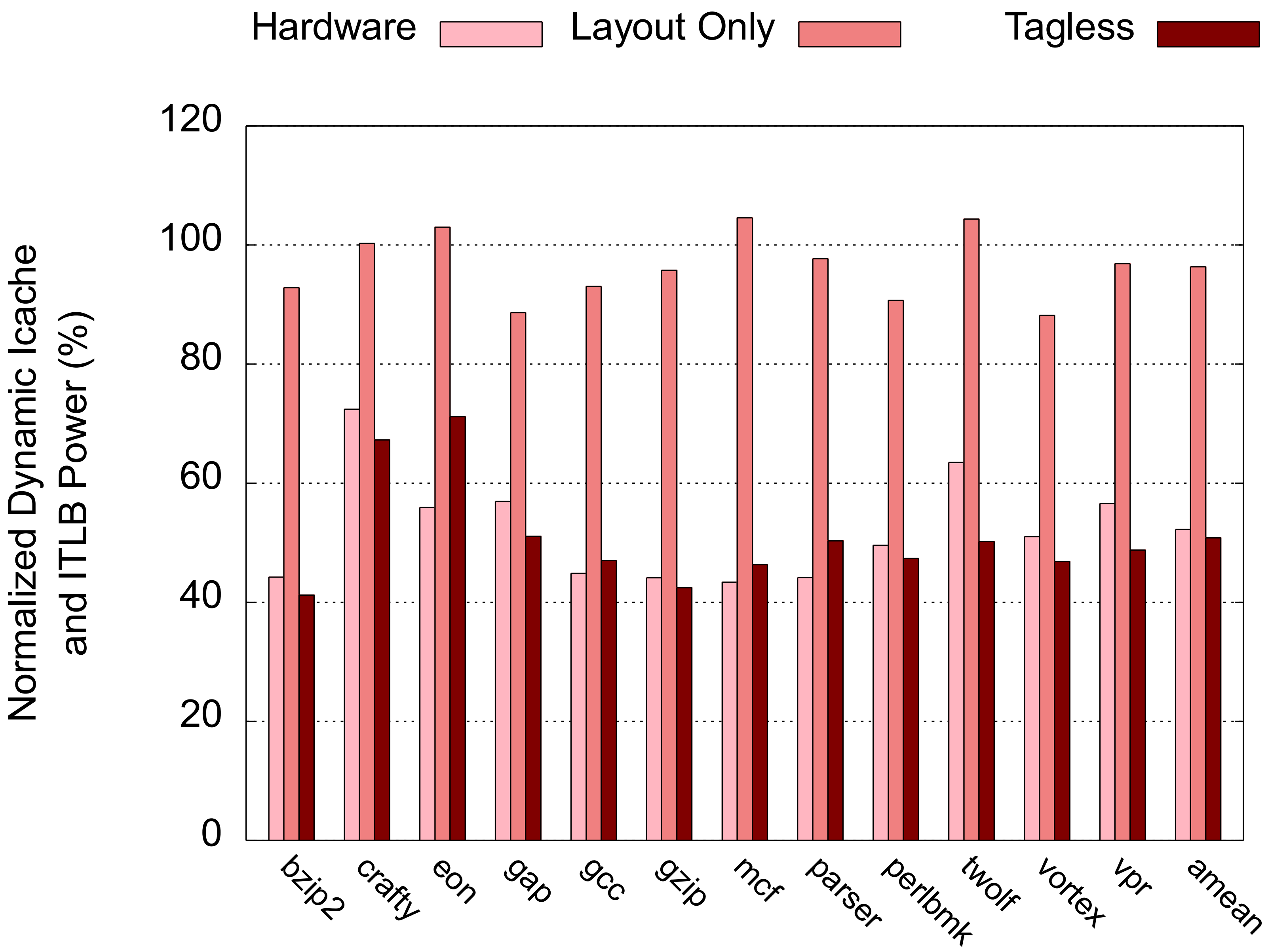
Evaluation

- **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

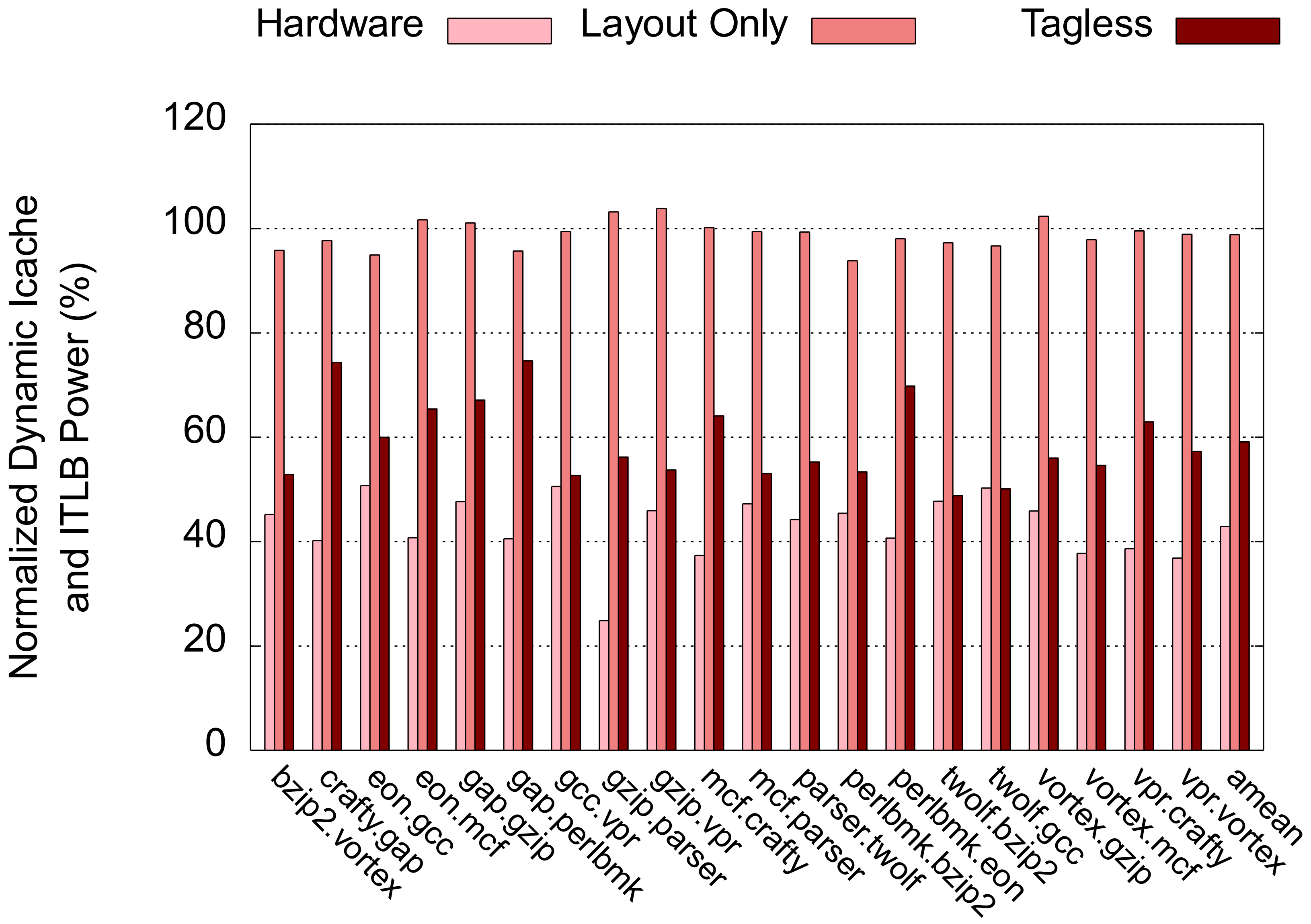


And Large Power Savings (49%)



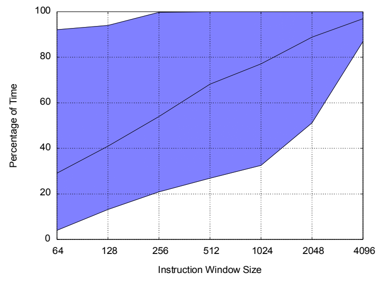
UNIVERSITY OF CAMBRIDGE

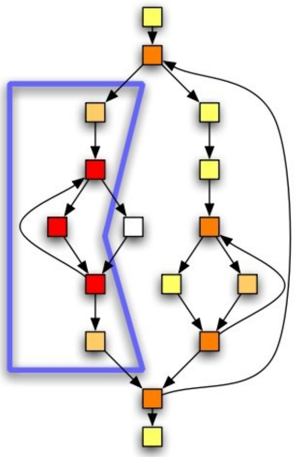
Power Savings with SMT (41%)

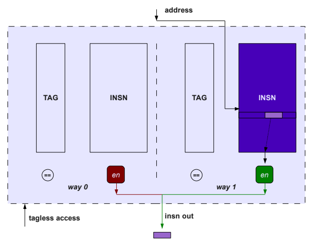


Conclusions

- **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