georgios.vavouliotis@bsc.es

Exploiting Page Table Locality for Agile TLB Prefetching

Georgios Vavouliotis[1,2] *Lluc Alvarez*[1,2] *Vasileios Karakostas*[3]
Konstantinos Nikas[3] *Nectarios Koziris*[3] *Daniel A. Jiménez*[4] *Marc Casas*[1,2]

[1]*Barcelona Supercomputing Center*
[2]*Universitat Politecnica de Catalunya*
[3]*National Technical University of Athens*
[4]*Texas A&M University*

Executive Summary

- Problem Statement
  - Address translation overheads due to data accesses
- Our approach —> TLB Prefetching
  - Operates on the microarchitectural level
  - Relies on the memory access patterns of the application
  - Does not disrupt the virtual memory subsystem
- Contributions
  - Sampling-based Free TLB Prefetching (SBFP)
    - Exploit page table locality to enhance TLB prefetching
  - Agile TLB Prefetcher (ATP)
    - Novel composite TLB prefetching scheme

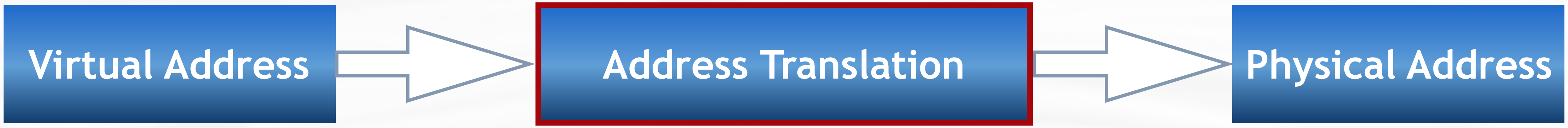
Combining ATP with SBFP improves geomean performance by more than 10% across different benchmark suites and reduces most of the page walk references to the memory hierarchy

Outline

- Background
- Sampling-based Free TLB Prefetching (SBFP)
- Agile TLB Prefetcher (ATP)
- Methodology
- Evaluation
- Conclusions

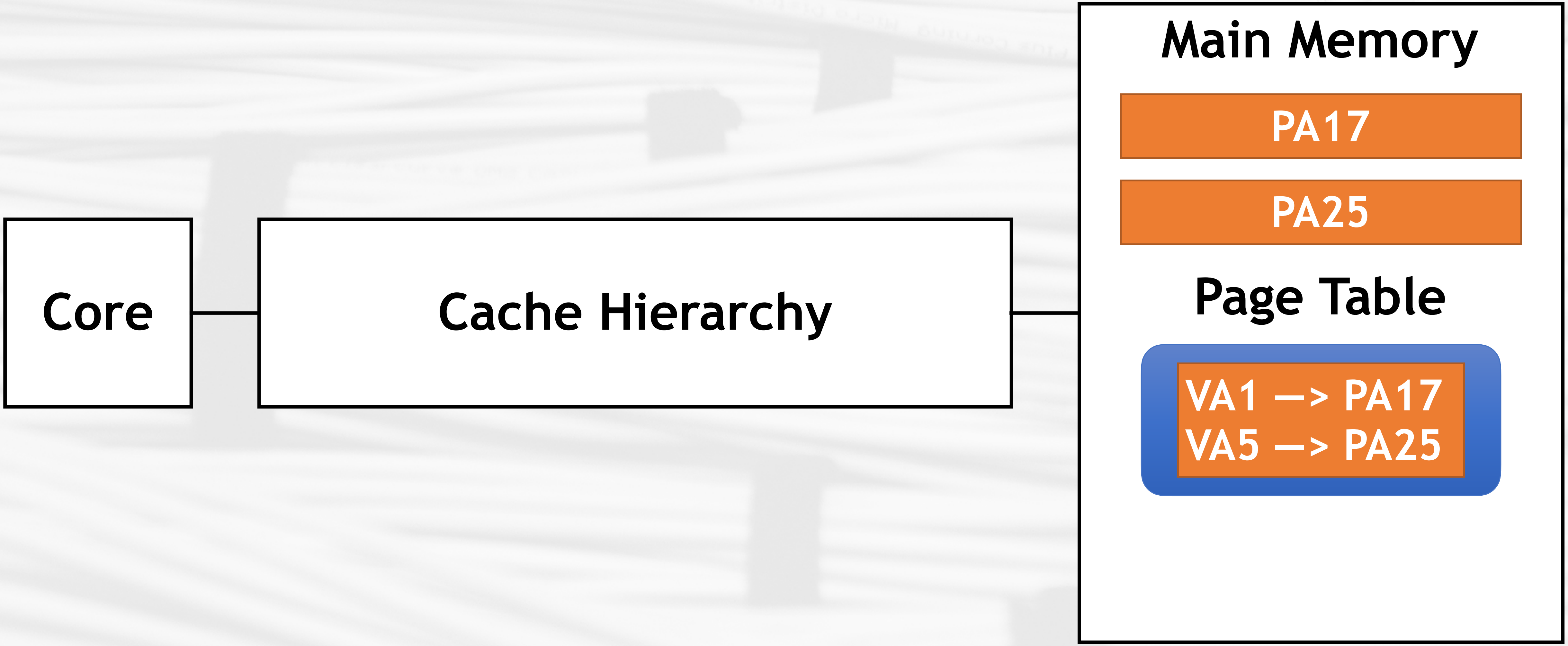
Address Translation

- Each memory access requires a virtual to physical translation

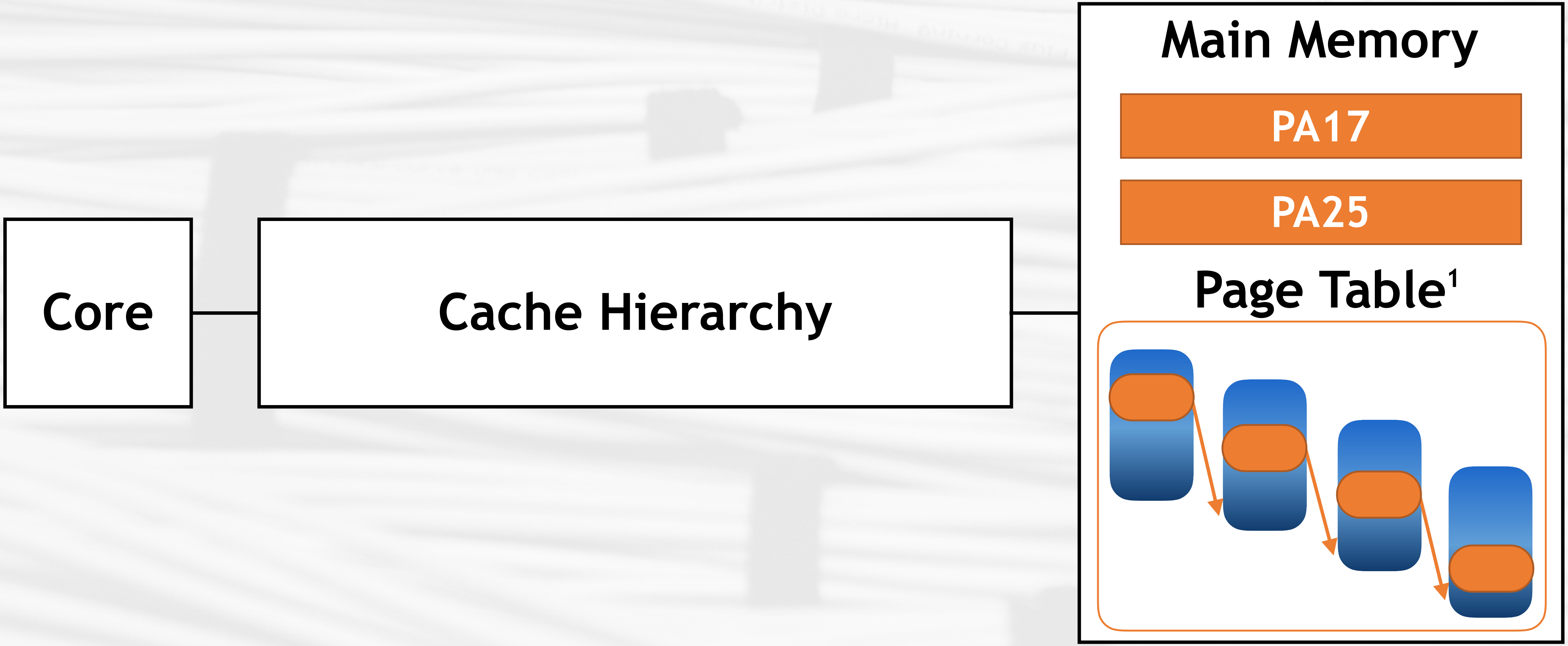


- Modern systems provide sophisticated software and hardware support to accelerate address translation

x86 Architectural Support for Address Translation



x86 Architectural Support for Address Translation



Core

Cache Hierarchy

Main Memory

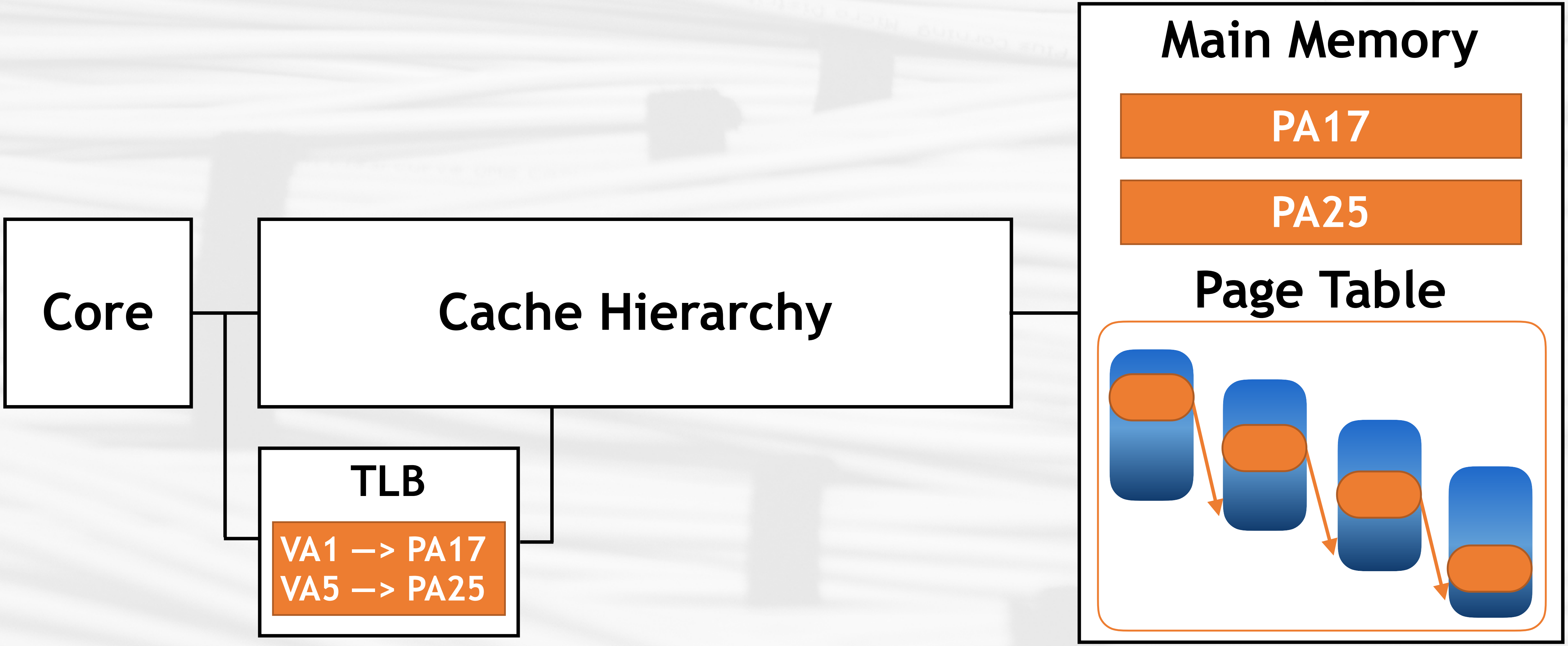
PA17

PA25

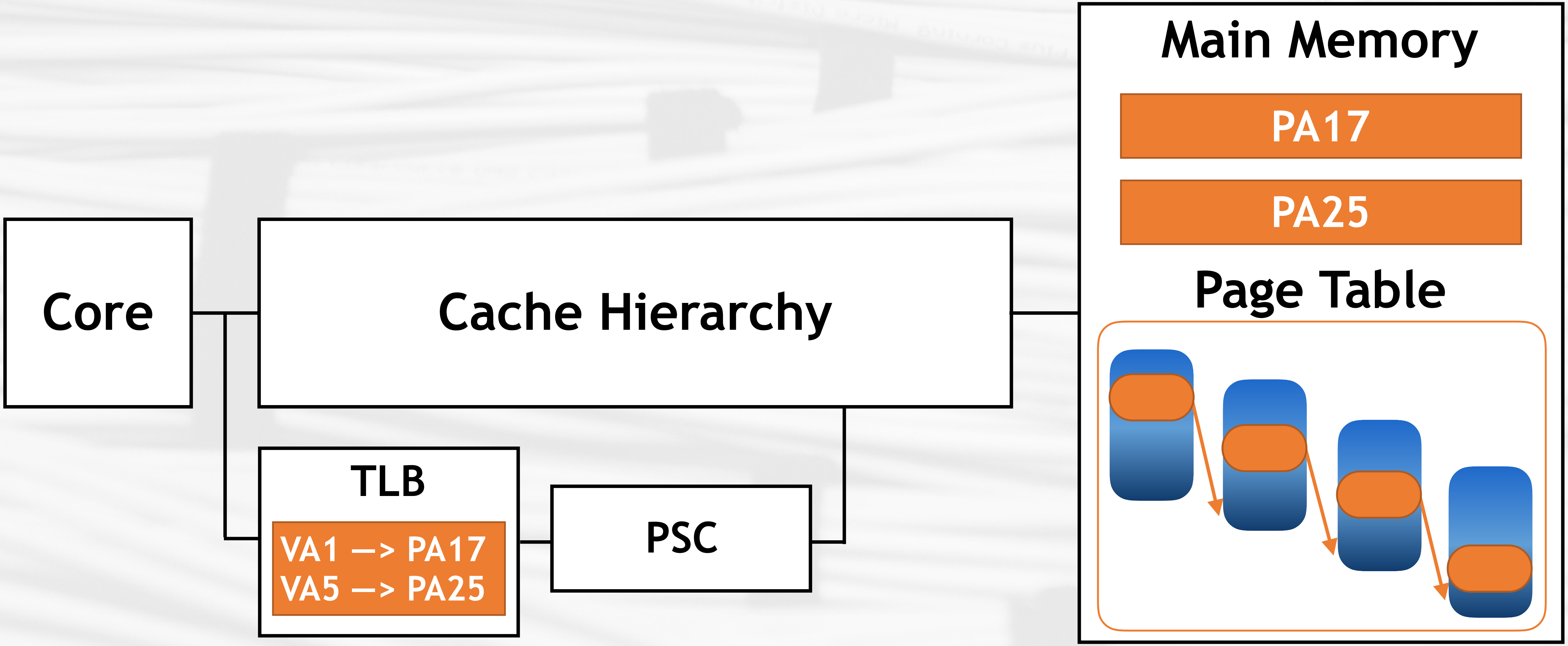
Page Table[1]

[1] Intel 5-level Paging, https://software.intel.com/content/www/us/en/develop/download/5-level-paging-and-5-level-ept-white-paper.html

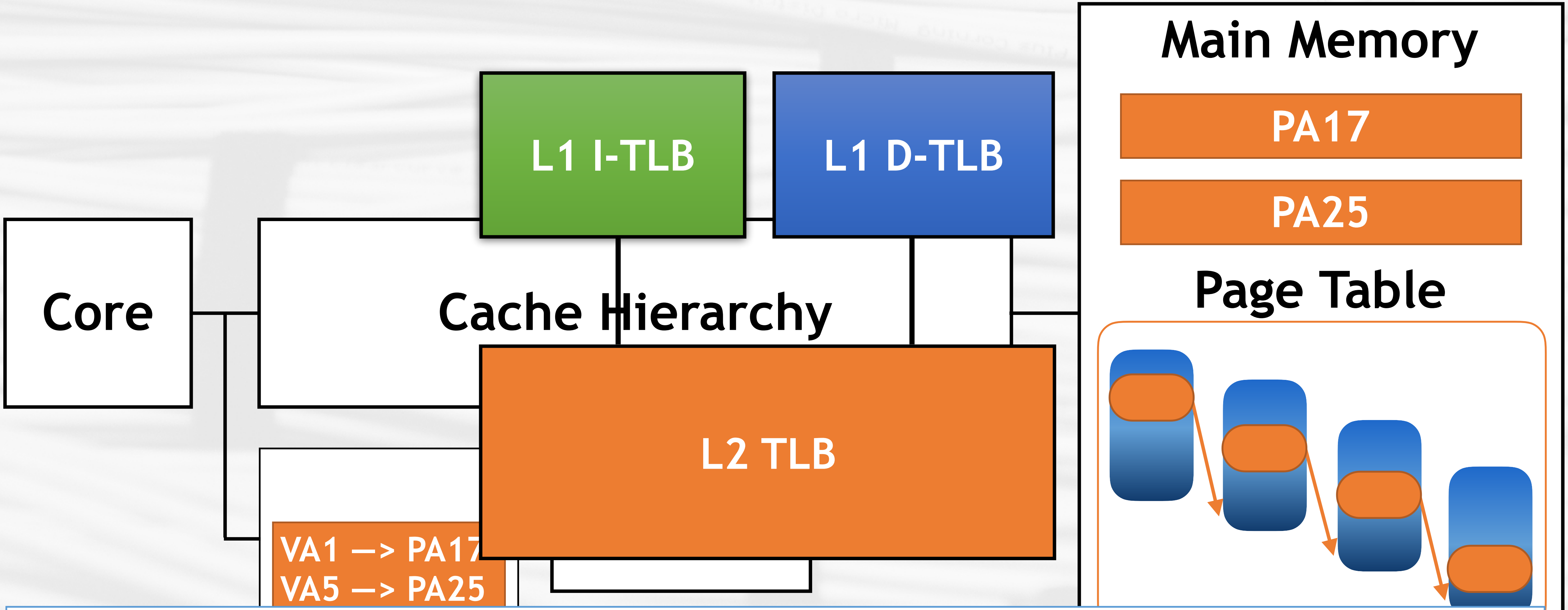
x86 Architectural Support for Address Translation



x86 Architectural Support for Address Translation



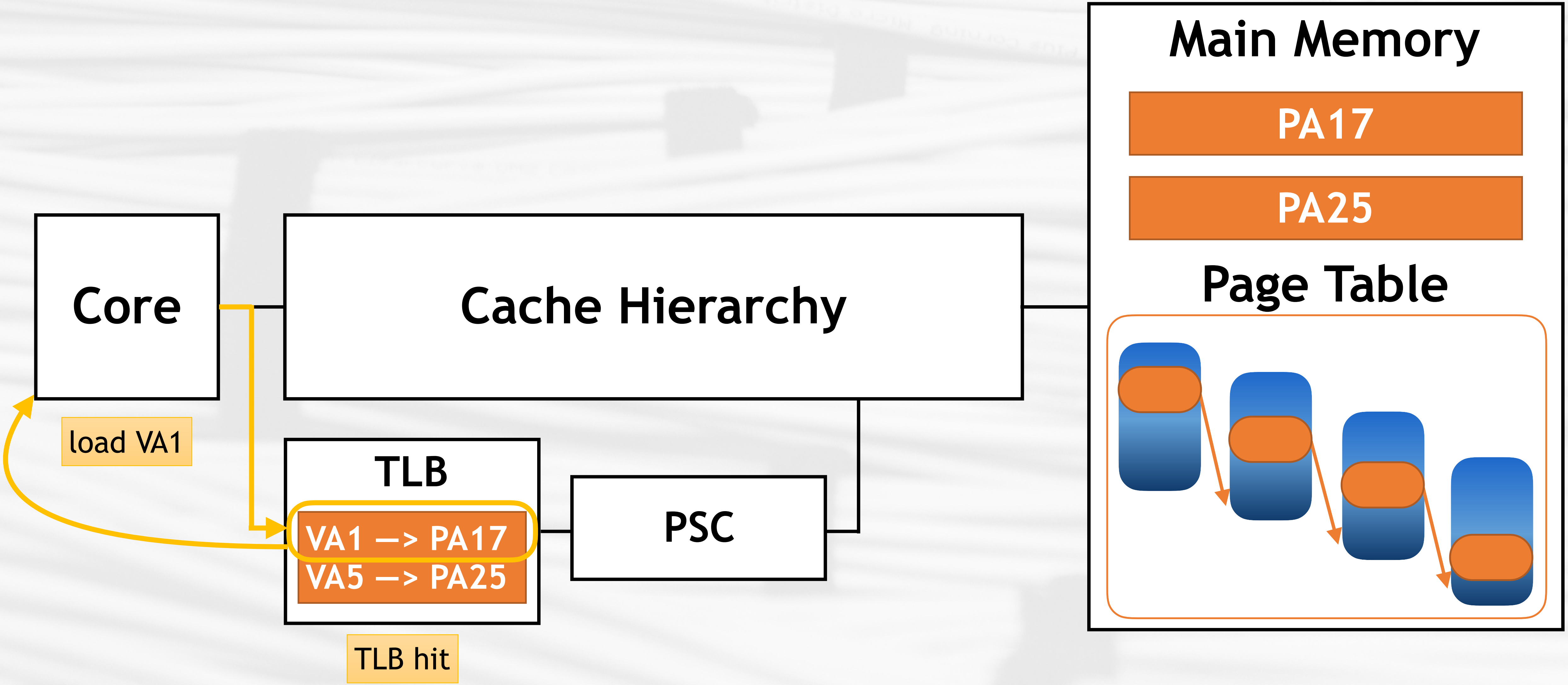
Translation Lookaside Buffer (TLB)



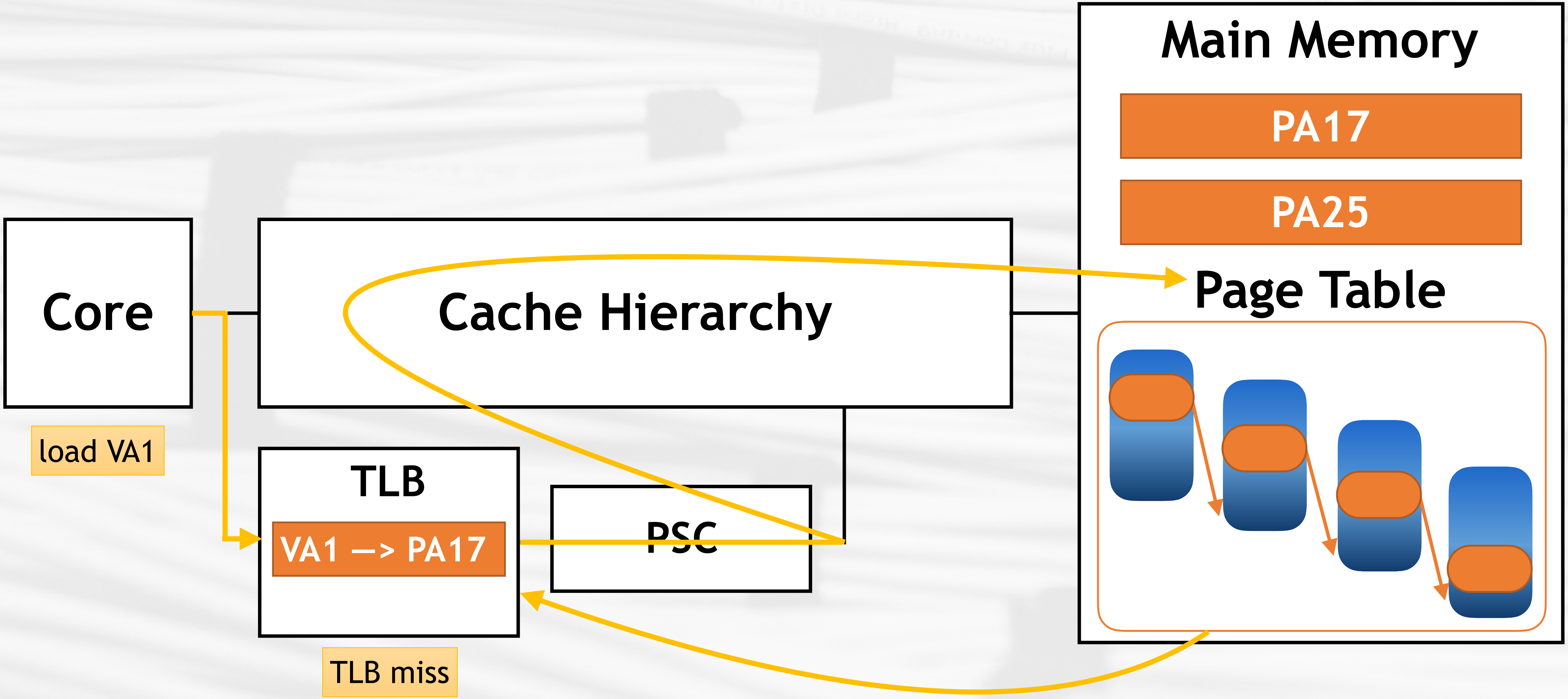
Last level TLB misses account for most of the cycles spent in the TLB miss handler[1]

[1] Basu et. al, "Efficient virtual memory for big memory servers", ISCA'13

Translation Lookaside Buffer (TLB)



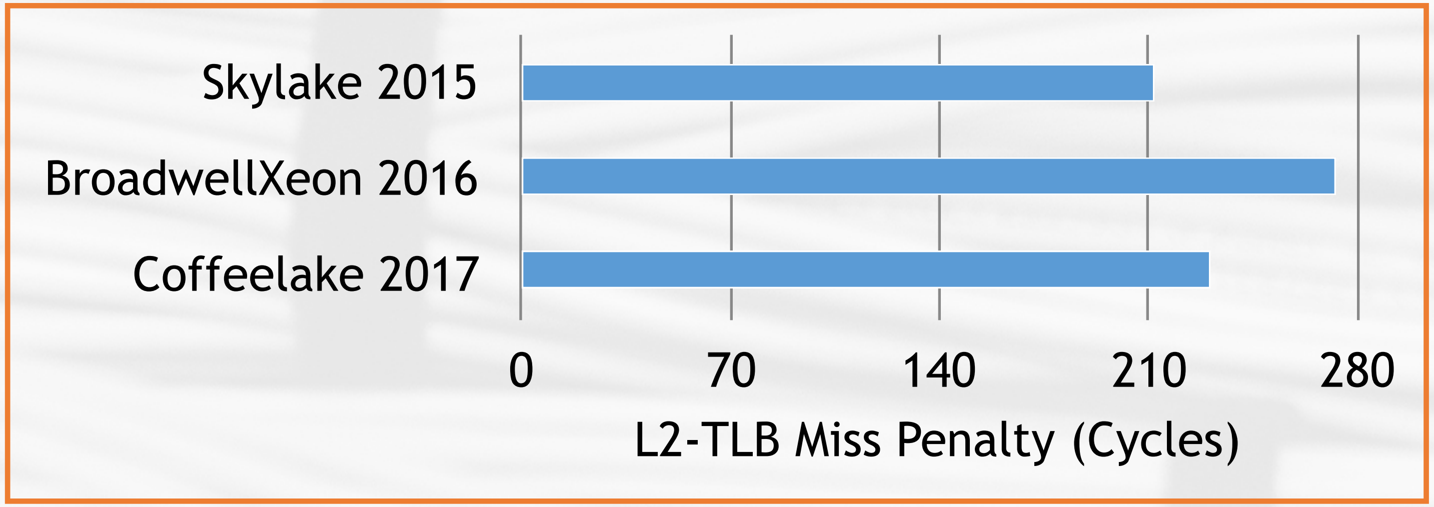
Translation Lookaside Buffer (TLB)



Translation Lookaside Buffer (TLB)

Address translation poses severe performance penalties, especially for big data applications[1,2]

**



Our analysis indicates that a Perfect TLB[3] improves geometric mean performance by up to 70% across different contemporary benchmark suites

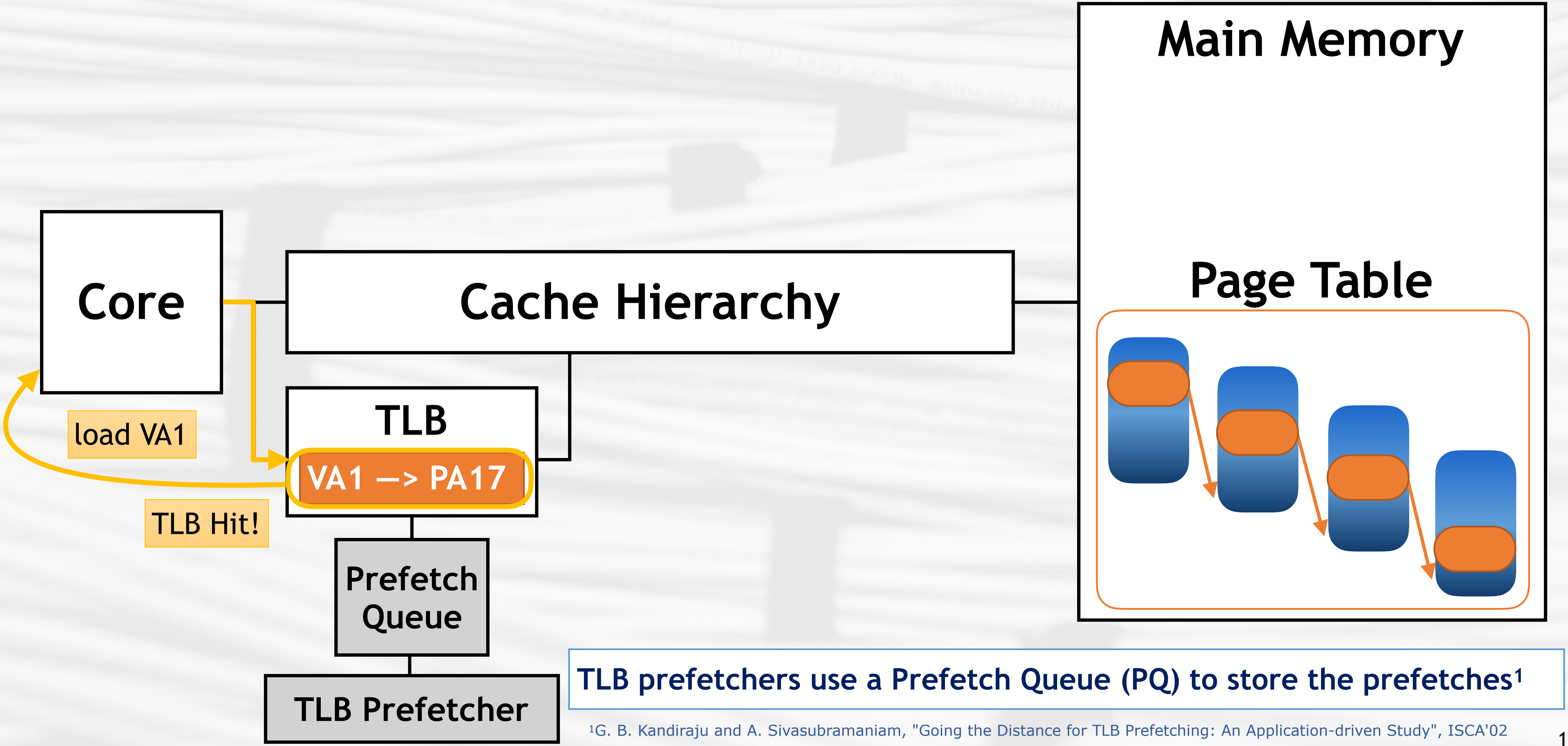
Address translation overhead is an emerging problem

**Gras et. al, "Translation Leak-aside Buffer: Defeating Cache Side-channel Protections with TLB Attacks", SEC'18

[1] XSBench, https://github.com/ANL-CESAR/XSBench [2] Beamer et. al, "The GAP benchmark suite", CoRR'15

[3] Perfect TLB = ideal TLB where all accesses are hits

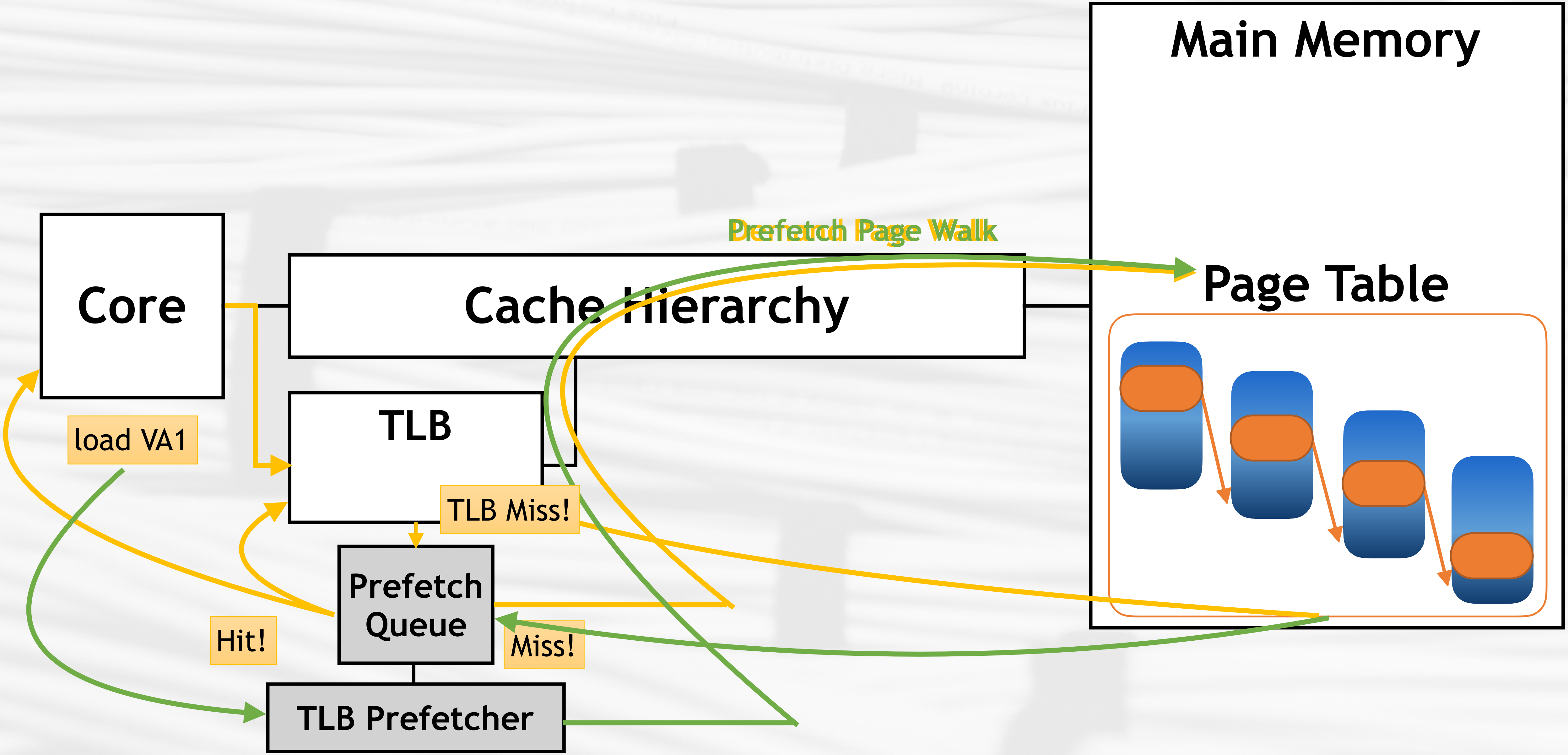
TLB Prefetching



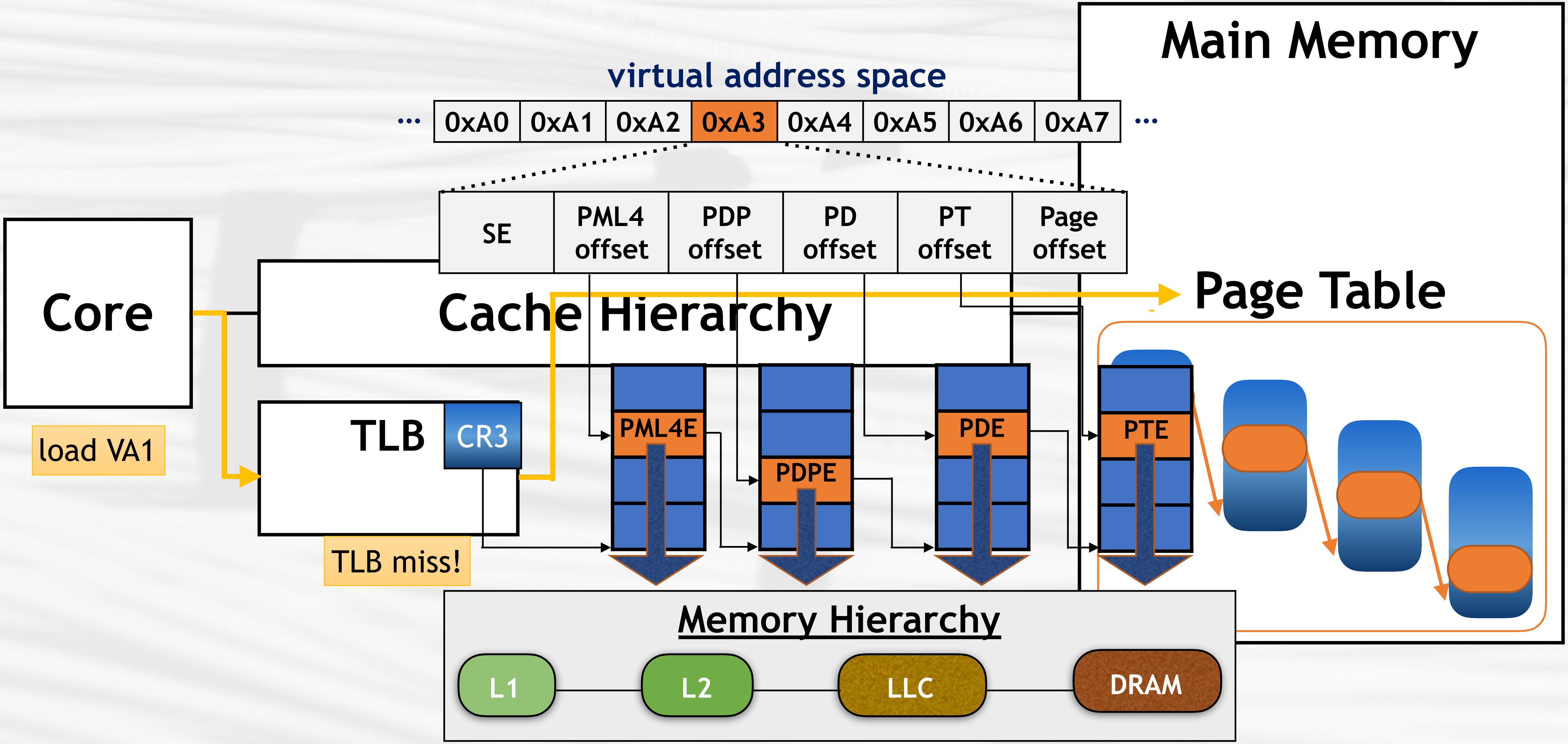
TLB prefetchers use a Prefetch Queue (PQ) to store the prefetches[1]

[1] G. B. Kandiraju and A. Sivasubramaniam, "Going the Distance for TLB Prefetching: An Application-driven Study", ISCA'02

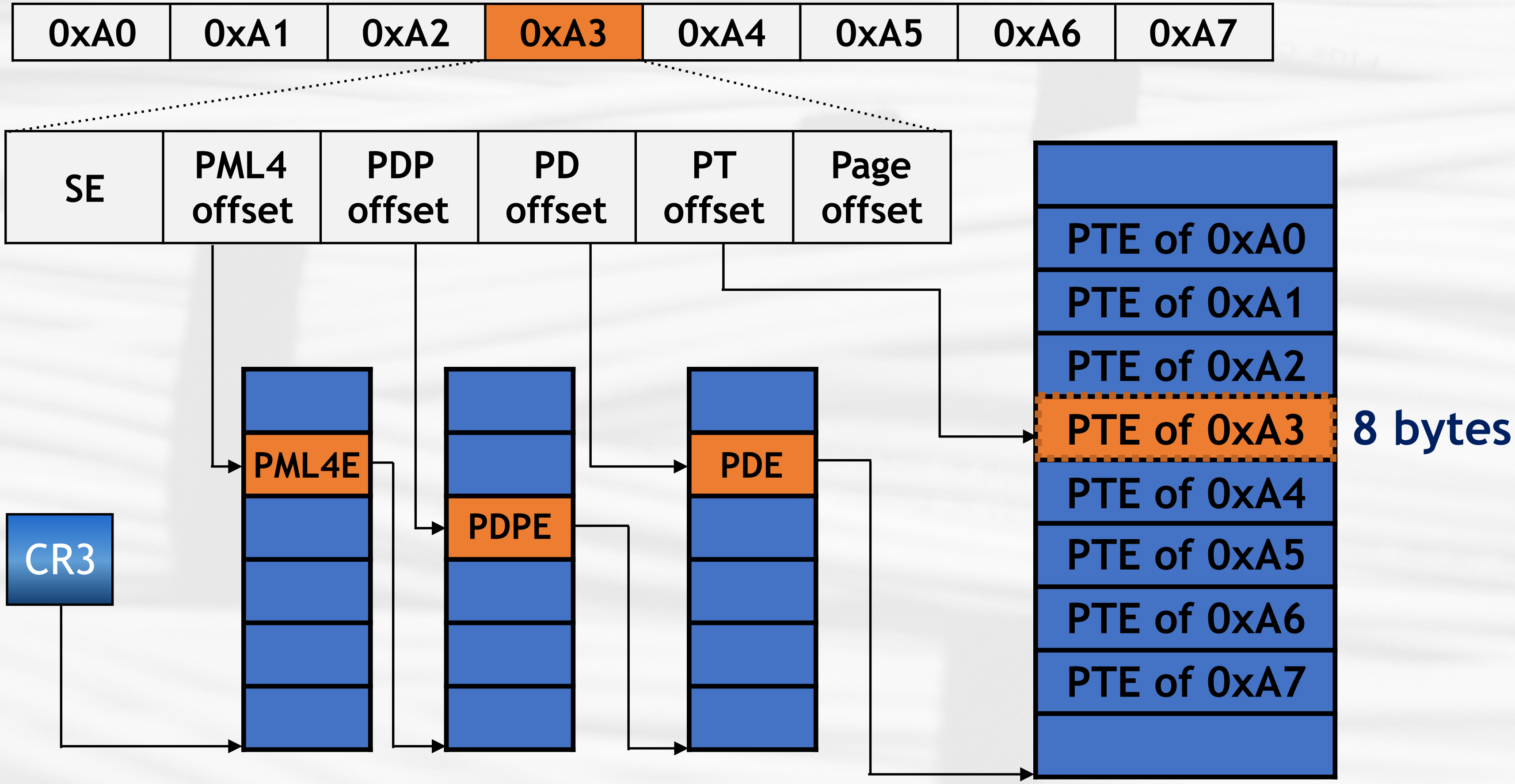
TLB Prefetching



x86 Page Table Walking



Motivation for Exploiting Page Table Locality



0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0xA6 0xA7

SE | PML4 offset | PDP offset | PD offset | PT offset | Page offset

CR3

PML4E

PDPE

PDE

PTE of 0xA0

PTE of 0xA1

PTE of 0xA2

PTE of 0xA3

8 bytes

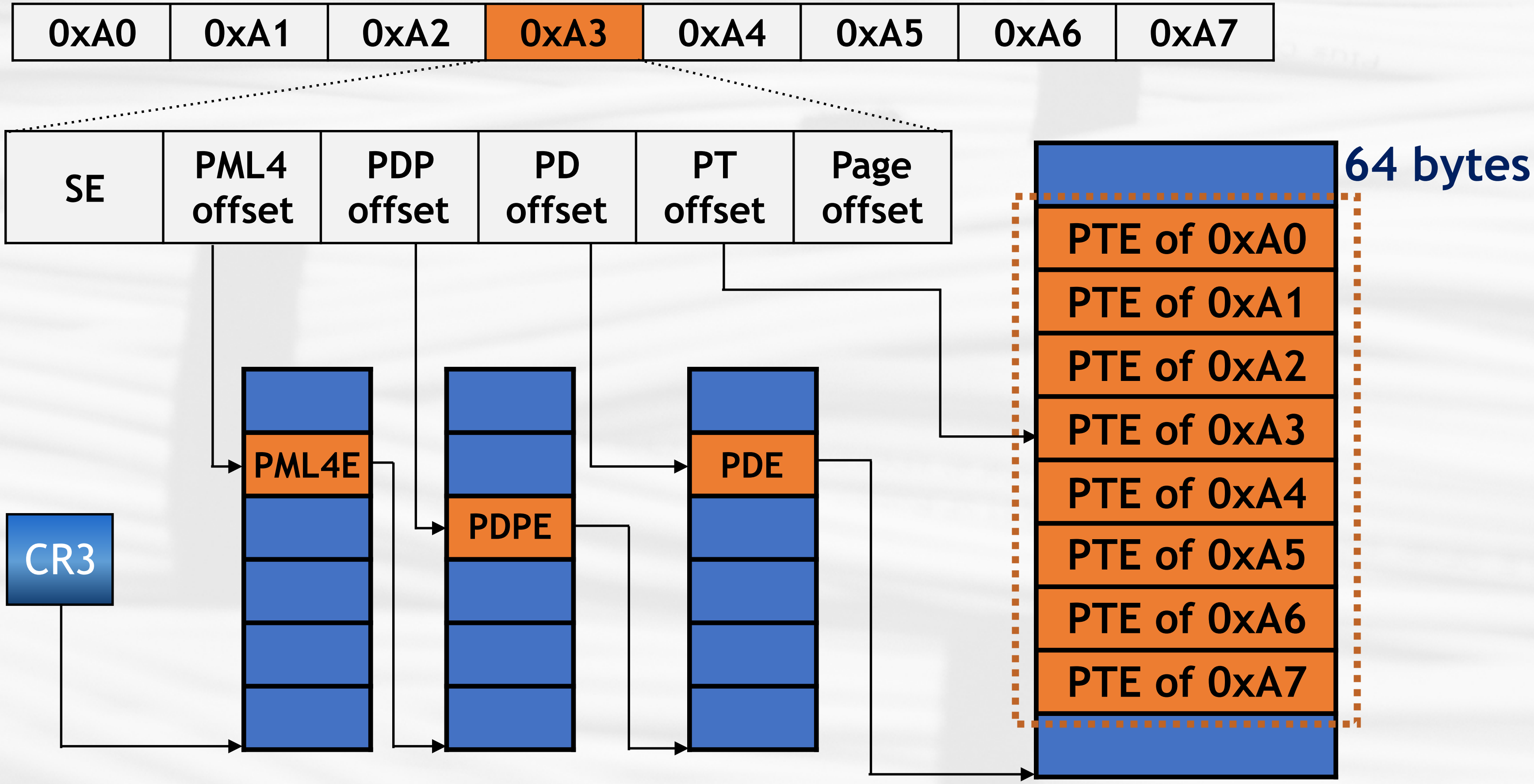
PTE of 0xA4

PTE of 0xA5

PTE of 0xA6

PTE of 0xA7

Motivation for Exploiting Page Table Locality



0xA0	0xA1	0xA2	0xA3	0xA4	0xA5	0xA6	0xA7

SE	PML4 offset	PDP offset	PD offset	PT offset	Page offset

CR3

PML4E

PDPE

PDE

64 bytes

PTE of 0xA0

PTE of 0xA1

PTE of 0xA2

PTE of 0xA3

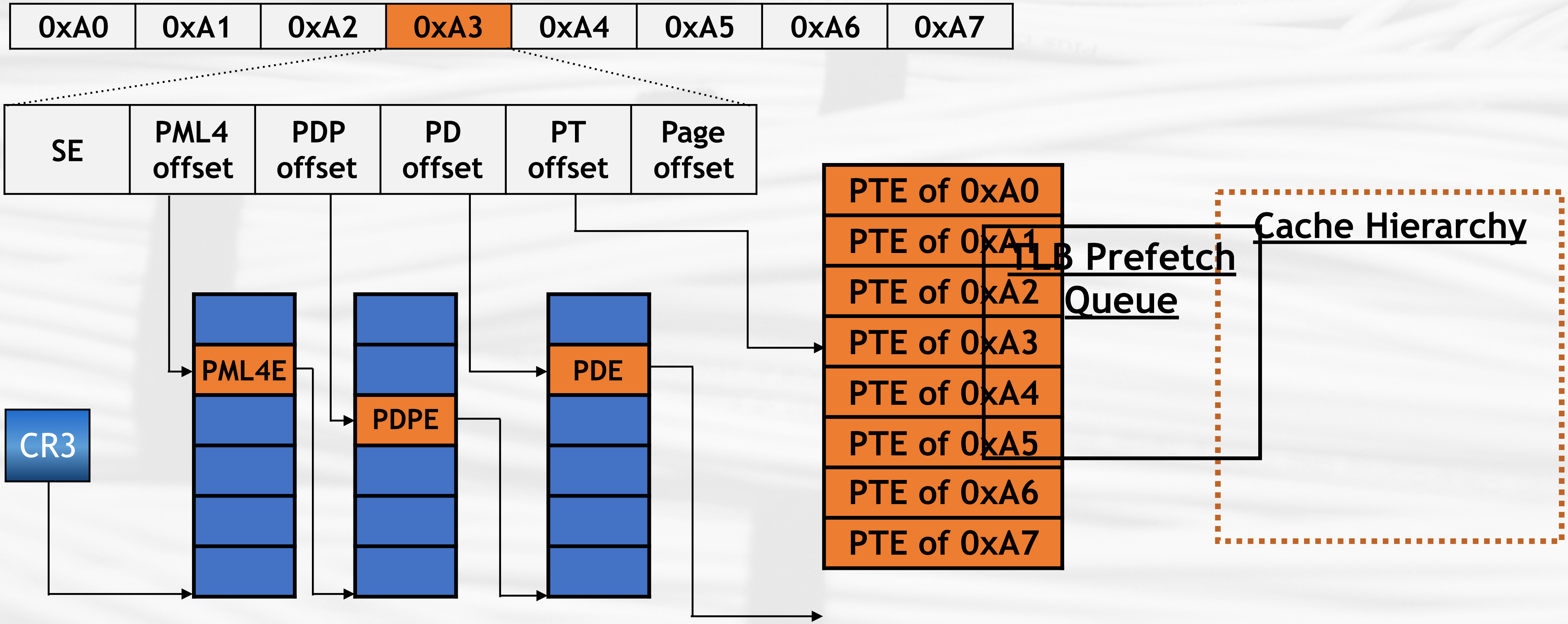
PTE of 0xA4

PTE of 0xA5

PTE of 0xA6

PTE of 0xA7

Motivation for Exploiting Page Table Locality



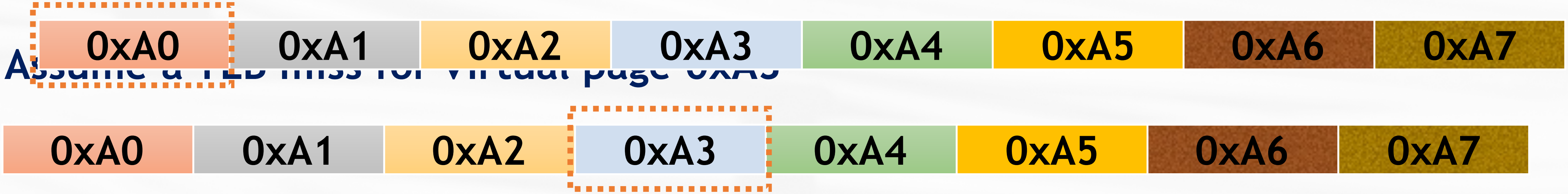
Prefetching all the free PTEs per page walk results in sub-optimal performance gains

Sampling-based Free TLB Prefetching (SBFP)

- Prefetch only the useful free PTEs per page walk
- Adapt to phase-based behaviors
- Can be combined with any TLB prefetcher
- Exploit page table locality for both demand and prefetch page walks

Terminology

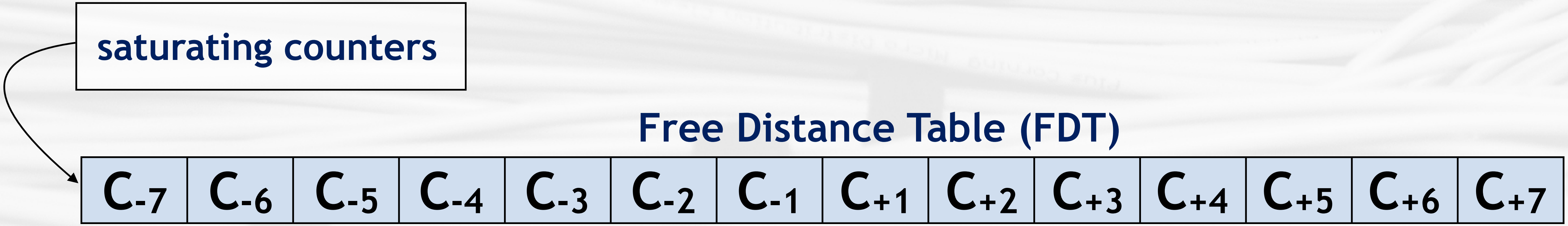
- After a page table walk, 8 PTEs are stored into a cache line
- Free Distance = signed distance between the PTE that holds the demand translation and another free PTE
  - 14 possible free distances, from -7 to +7 (excluding zero)
- Assume a TLB miss for virtual page 0xA3



cache_position(0xA3) = 3

e.g., Free Distance(0xA0) = -3

Sampling-based Free TLB Prefetching — Overview



saturating counters

Free Distance Table (FDT)

C_{-7}	C_{-6}	C_{-5}	C_{-4}	C_{-3}	C_{-2}	C_{-1}	C_{+1}	C_{+2}	C_{+3}	C_{+4}	C_{+5}	C_{+6}	C_{+7}

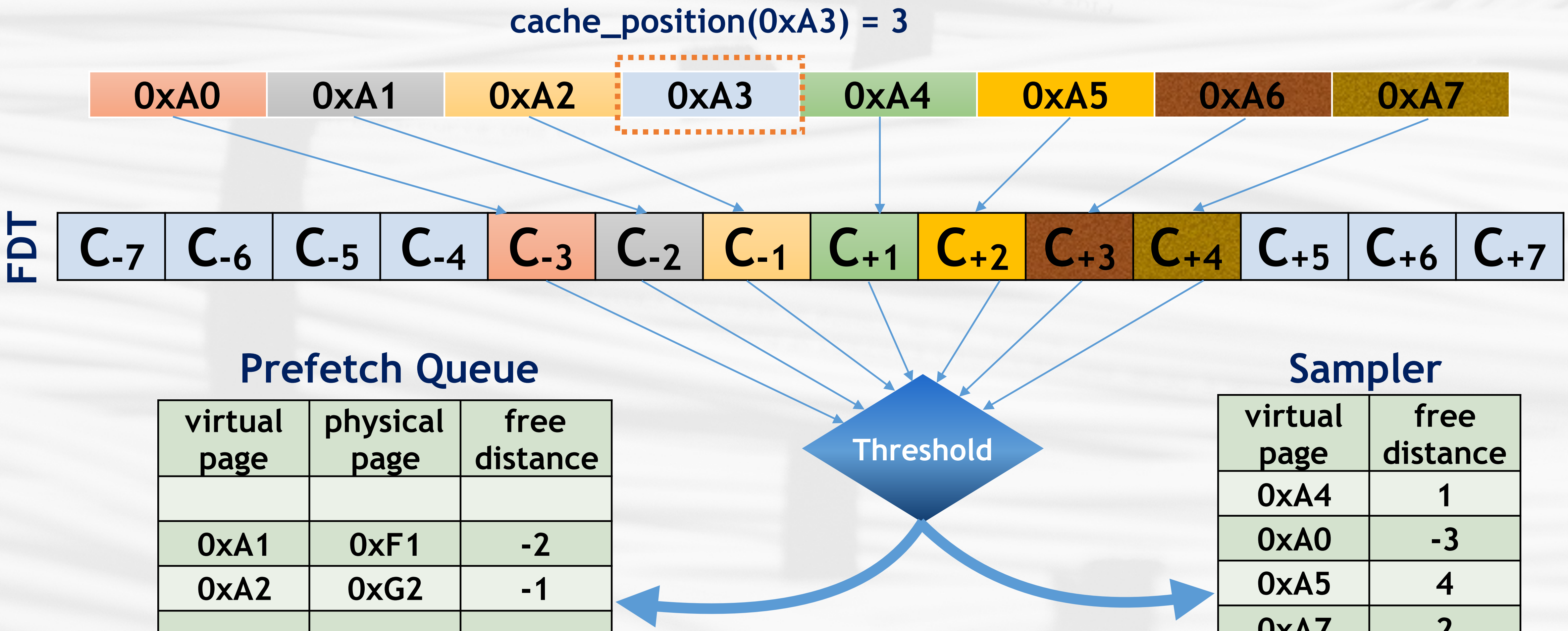
Prefetch Queue

virtual page	physical page	free distance

Sampler

virtual page	free distance

Sampling-based Free TLB Prefetching — Operation



cache_position(0xA3) = 3

0xA0	0xA1	0xA2	0xA3	0xA4	0xA5	0xA6	0xA7

FDT	C_{-7}	C_{-6}	C_{-5}	C_{-4}	C_{-3}	C_{-2}	C_{-1}	C_{+1}	C_{+2}	C_{+3}	C_{+4}	C_{+5}	C_{+6}	C_{+7}

Prefetch Queue

virtual page	physical page	free distance
0xA1	0xF1	-2
0xA2	0xG2	-1
0xA6	0xF6	3

Sampler

virtual page	free distance
0xA4	1
0xA0	-3
0xA5	4
0xA7	2

Sampling-based Free TLB Prefetching – Updating FDT

Free Distance Table (FDT)

C_{-7}	C_{-6}	C_{-5}	C_{-4}	C_{-3}	C_{-2}	C_{-1}	C_{+1}	C_{+2}	C_{+3}	C_{+4}	C_{+5}	C_{+6}	C_{+7}

Prefetch Queue

virtual page	physical page	free distance
0xA1	0xF1	-2
0xA2	0xG2	-1
0xA6	0xF6	3

Sampler

virtual page	free distance
0xA4	1
0xA0	-3
0xA5	4
0xA7	2

Sampling-based Free TLB Prefetching – Updating FDT

Free Distance Table (FDT)

C_{-7}	C_{-6}	C_{-5}	C_{-4}	C_{-3}	C_{-2}	C_{-1}	C_{+1}	C_{+2}	C_{+3}	C_{+4}	C_{+5}	C_{+6}	C_{+7}
					++			++					

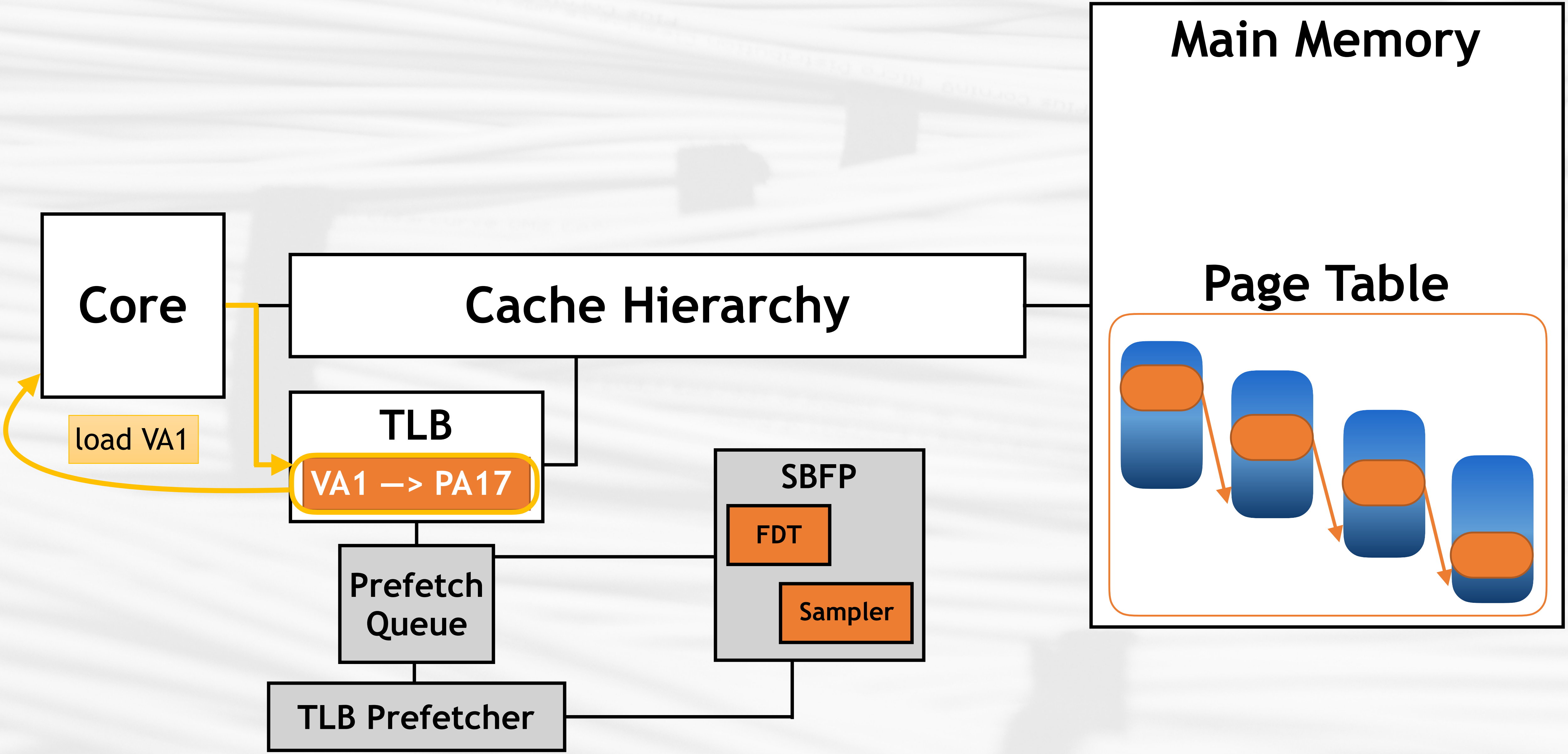
Prefetch Queue

virtual page	physical page	free distance
0xA1	0xF1	-2
0xA2	0xG2	-1
0xA6	0xF6	3

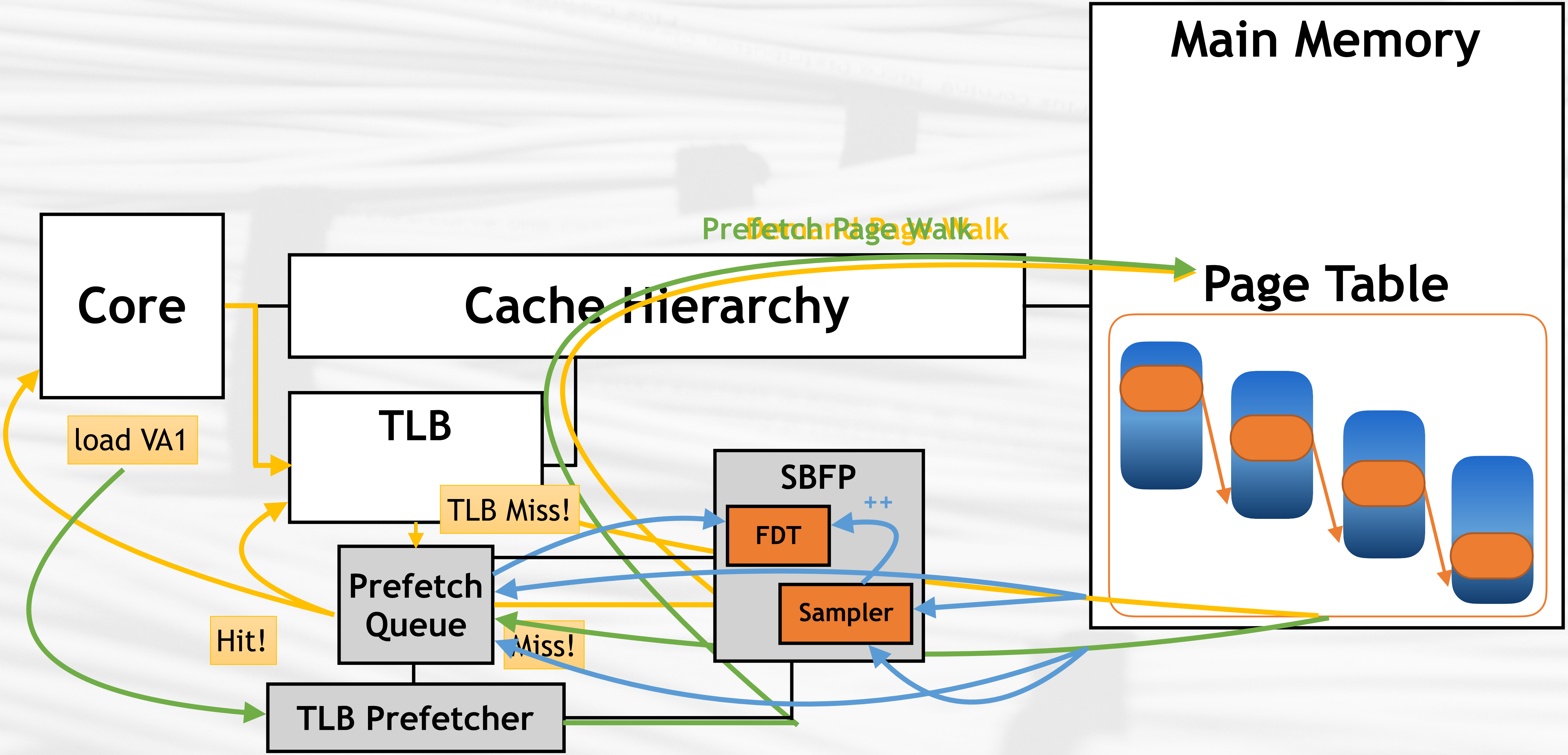
Sampler

virtual page	free distance
0xA4	1
0xA0	-3
0xA5	4
0xA7	2

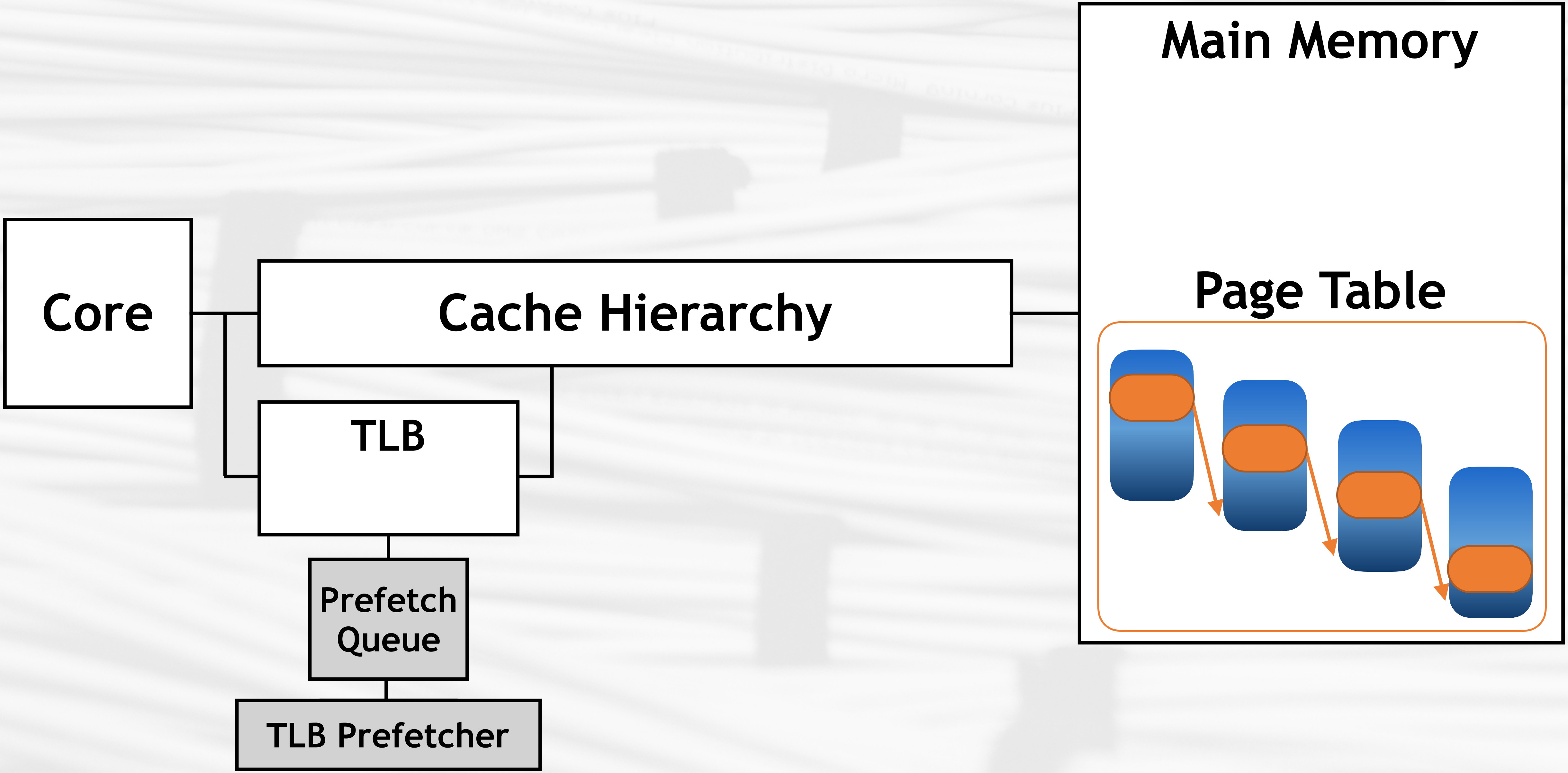
Combining SBFP with a TLB Prefetcher



Combining SBFP with a TLB Prefetcher



Agile TLB Prefetcher (ATP)



Motivation for Agile TLB Prefetcher (ATP)

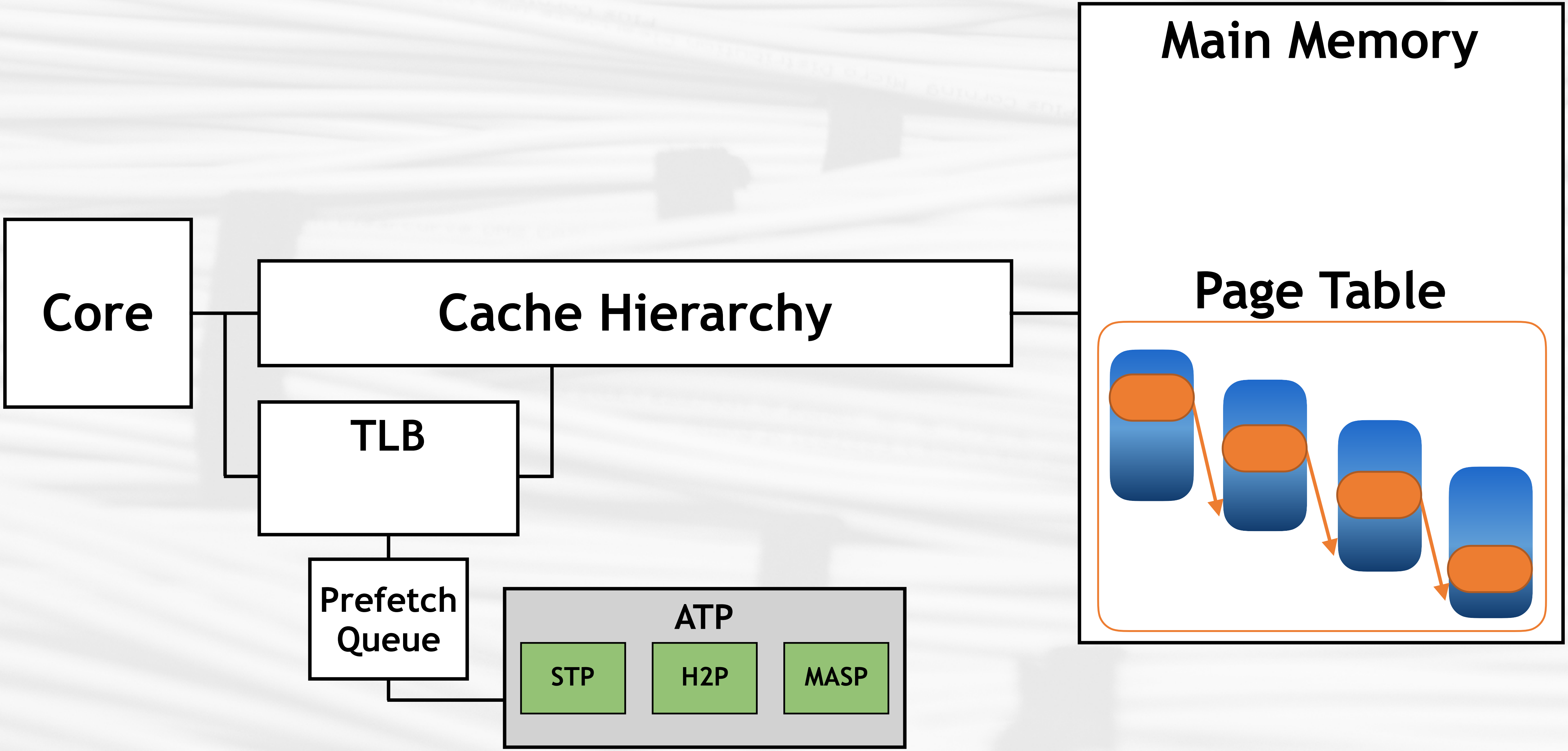
- **There is no state-of-the-art TLB prefetcher that performs best across all considered workloads**
- **Different workloads correlate well with different features (e.g., strides, distances between virtual pages, PC)**
- **When the TLB miss behavior is irregular, state-of-the-art TLB prefetchers issue useless prefetch requests**

Agile TLB Prefetcher (ATP)

- Combine three low-cost TLB prefetchers

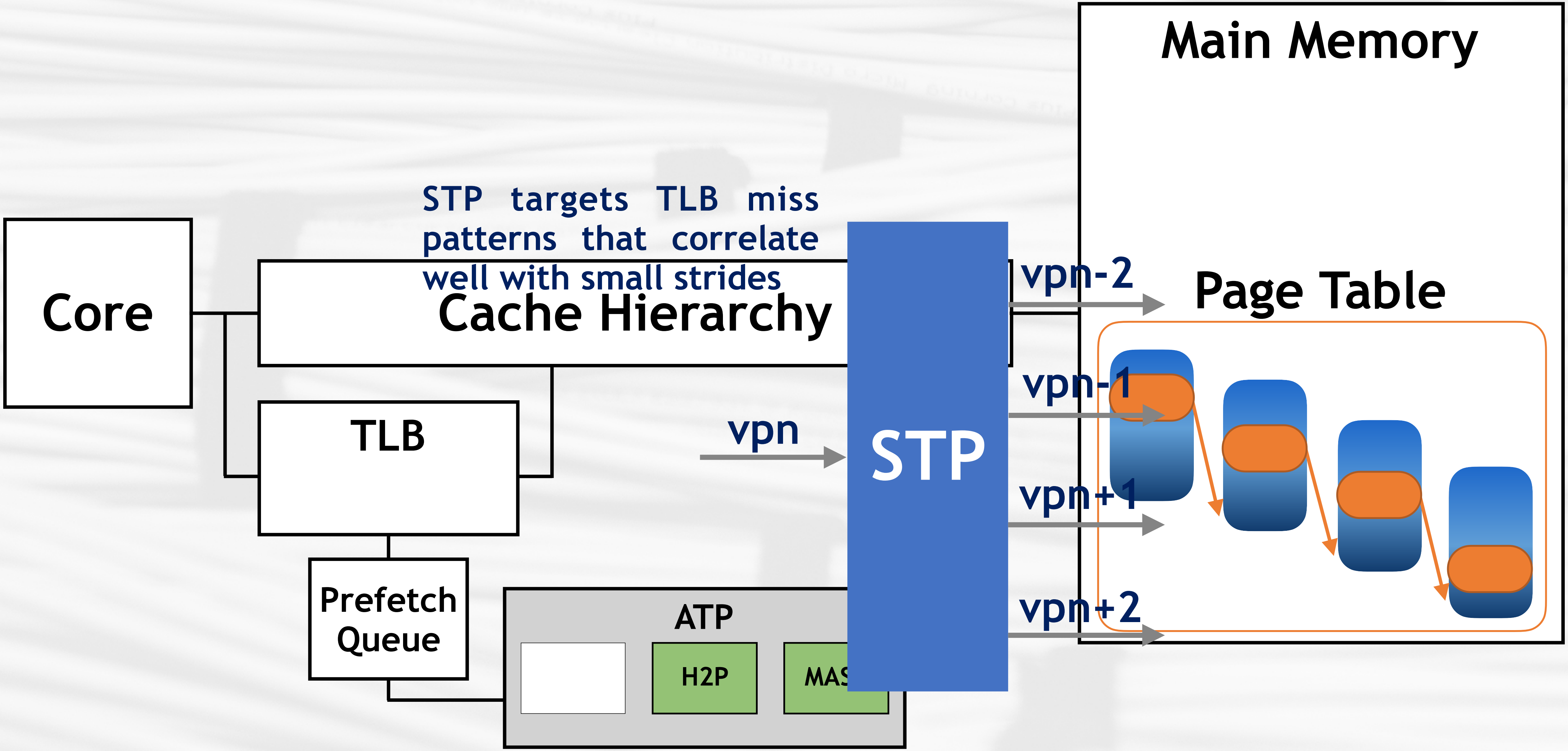
- Adaptive selection logic and throttling mechanisms

Constituent Prefetchers of ATP

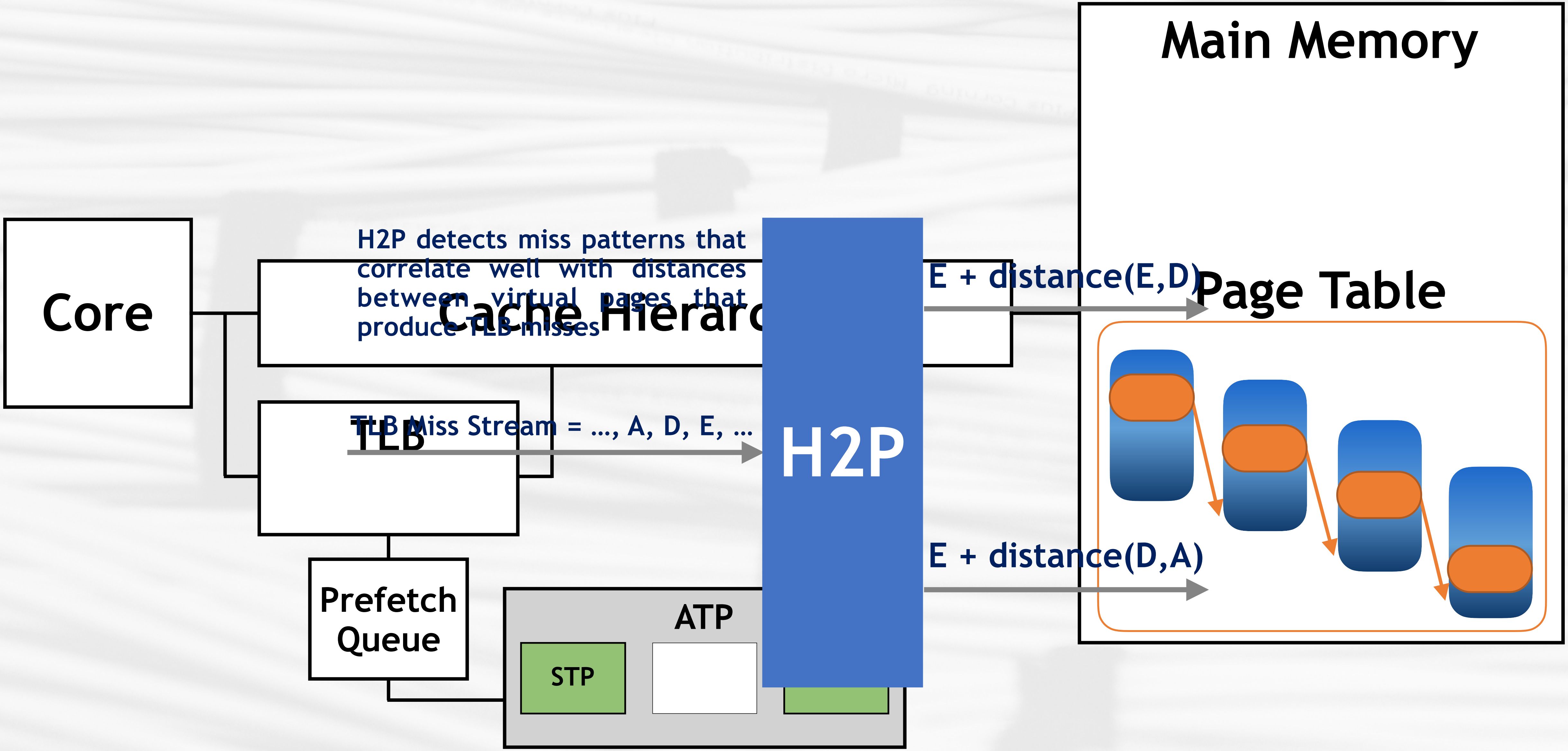


Stride TLB Prefetcher (STP)

STP targets TLB miss patterns that correlate well with small strides



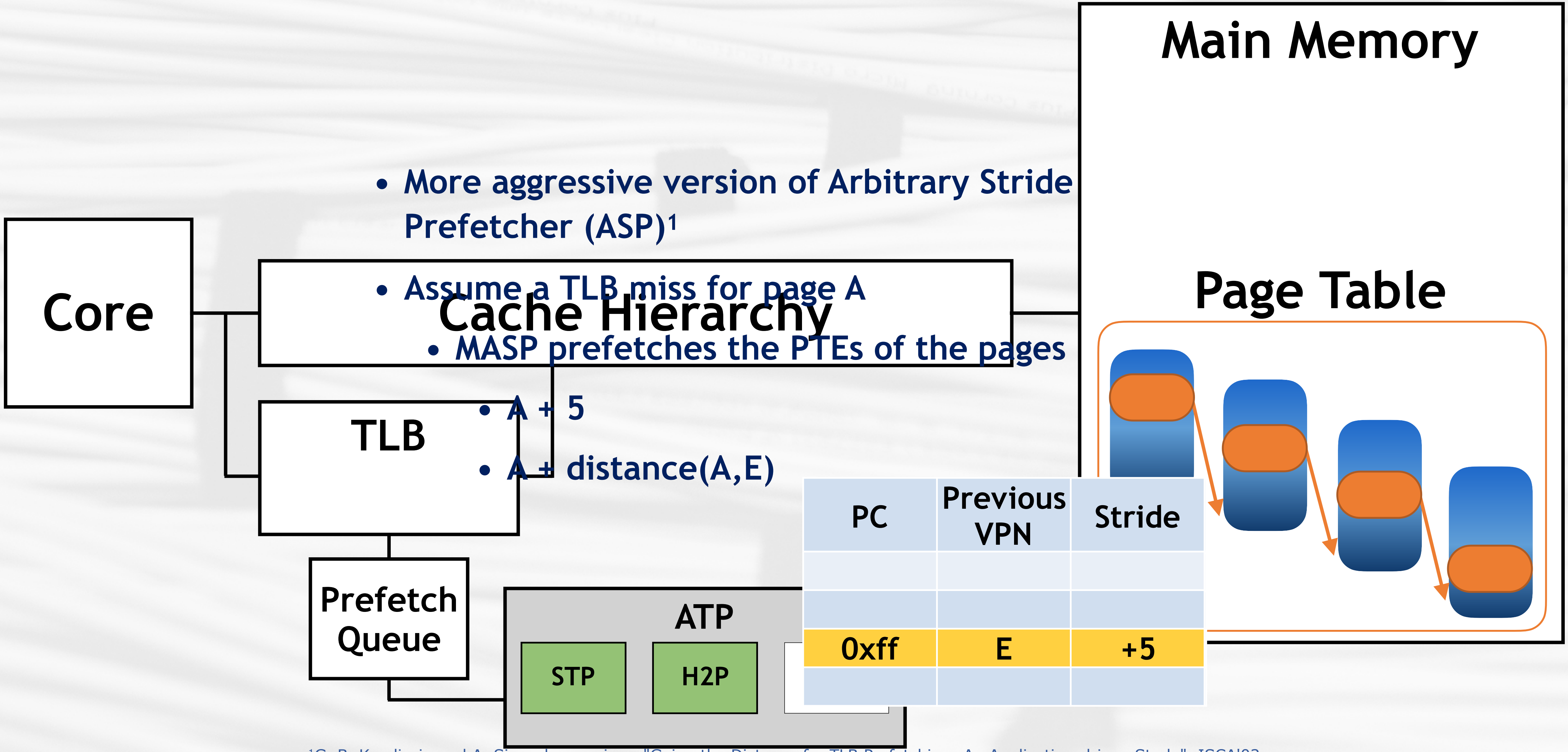
H2 Prefetcher (H2P)



H2P detects miss patterns that correlate well with distances between virtual pages that produce TLB misses

Modified Arbitrary Stride Prefetcher (MASP)

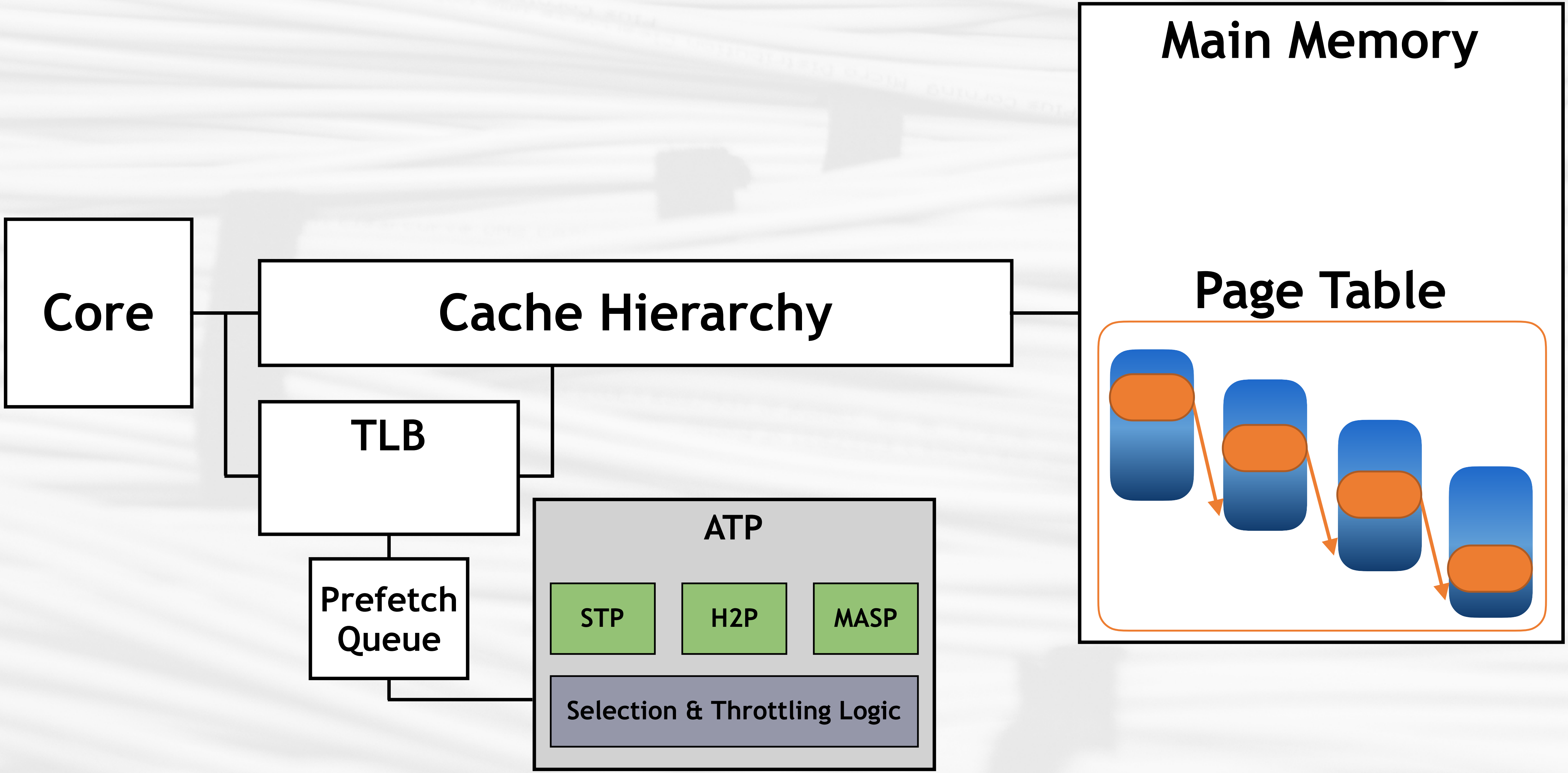
- More aggressive version of Arbitrary Stride Prefetcher (ASP)[1]
- Assume a TLB miss for page A
  - MASP prefetches the PTEs of the pages
    - A + 5
    - A + distance(A,E)



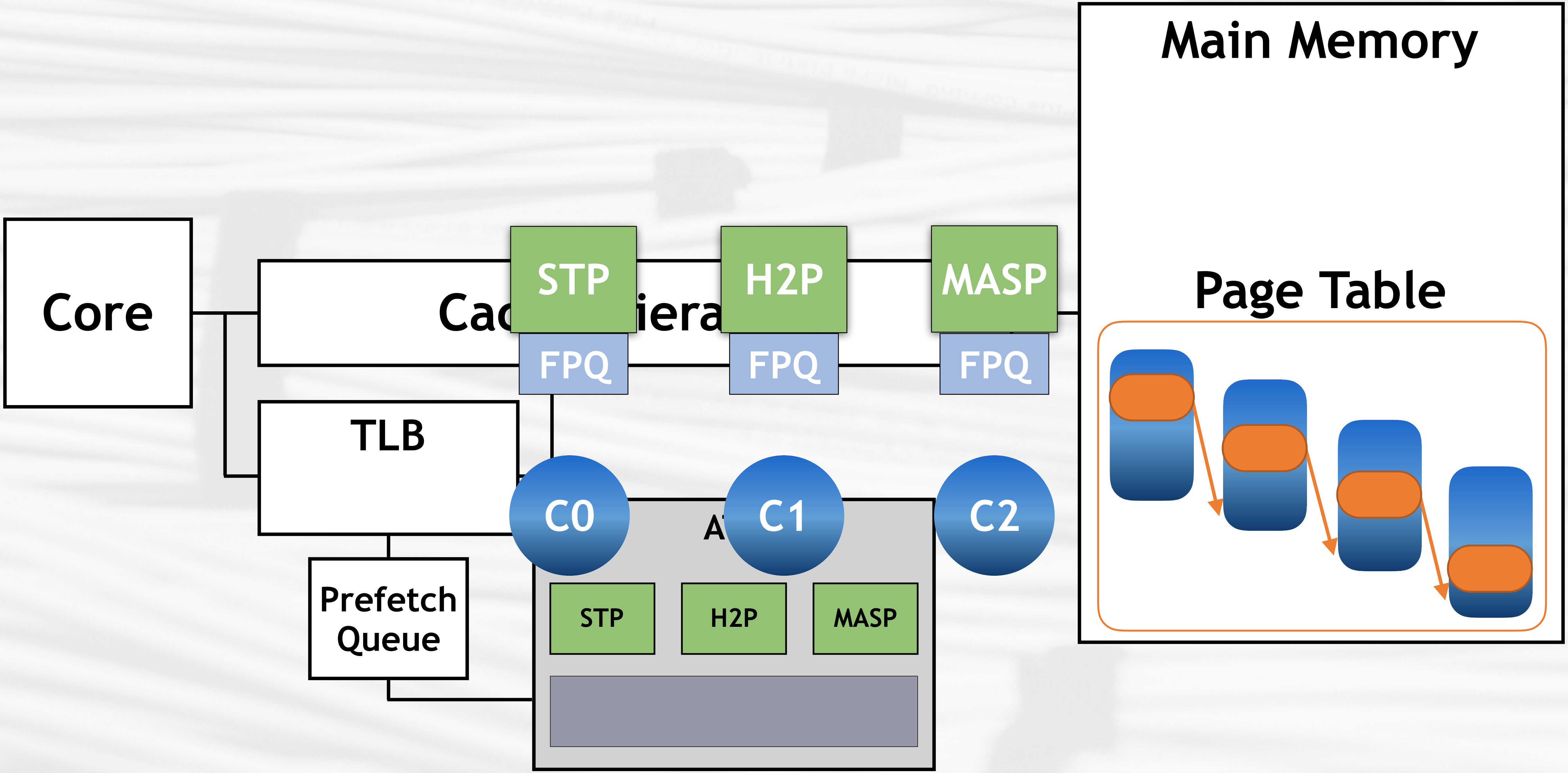
PC	Previous VPN	Stride
0xff	E	+5

[1] G. B. Kandiraju and A. Sivasubramaniam, "Going the Distance for TLB Prefetching: An Application-driven Study", ISCA'02

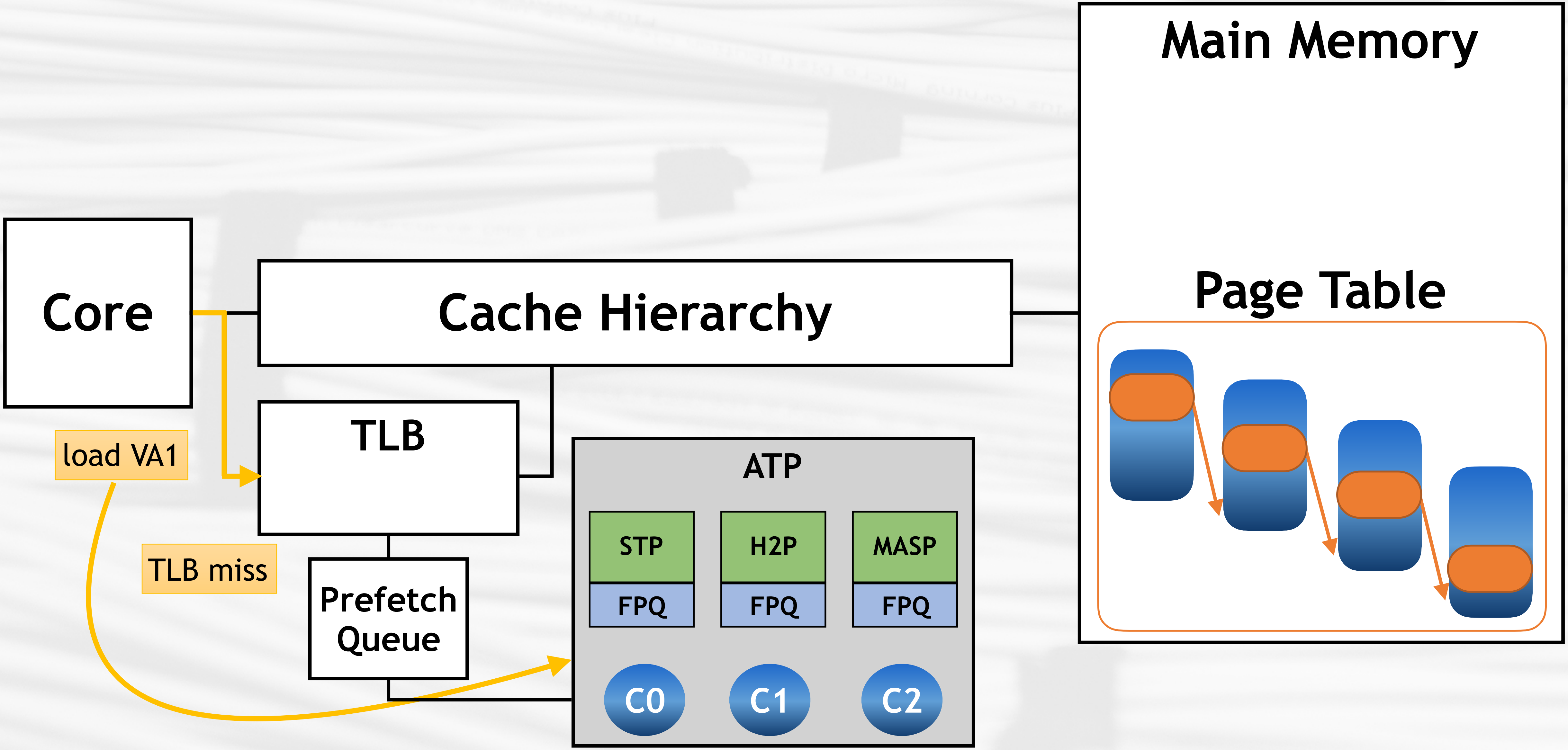
Selection and Throttling Schemes



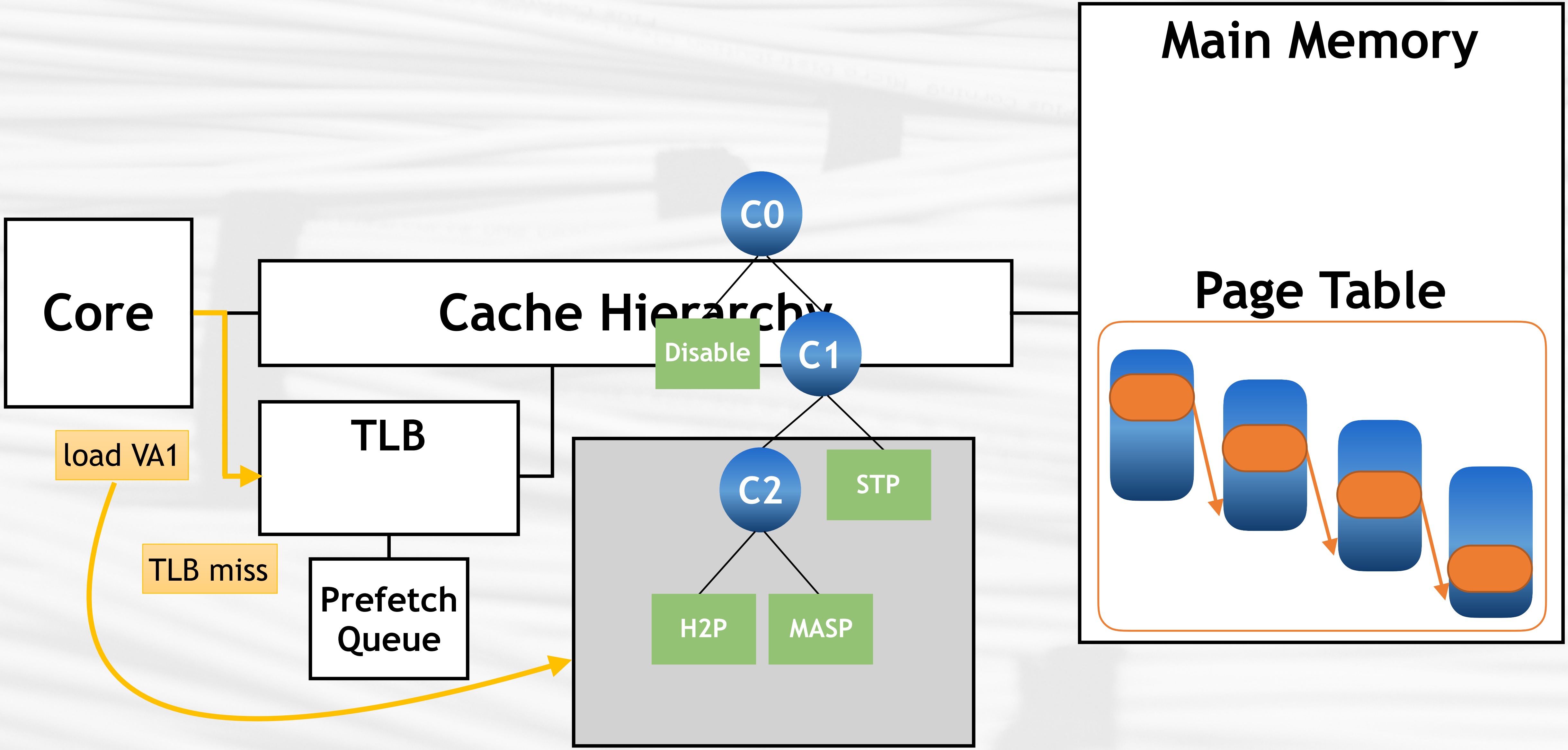
Selection and Throttling Schemes



Agile TLB Prefetcher (ATP) — Operation



Agile TLB Prefetcher (ATP) — Operation



Simulation Infrastructure

- ChampSim[1]
  - Trace-driven multi-core out-of-order simulator
  - x86 page table walker

Component	Parameters	Latency
L1 I-TLB	64-entry, 8-way	1cc
L1 D-TLB	64-entry, 8-way	1cc
L2 TLB	1536-entry, 12-way	8cc
Page Structure Caches	3-level Split PSC, PML4: 2-entry, PDP: 4-entry, PD: 32-entry	2cc
Prefetch Queue	64-entry, fully assoc.	2cc
Sampler	64-entry, fully assoc	2cc
L1 iCache	32KB, 8-way	1cc
L1 dCache	32KB, 8-way	1cc
L2 Cache	256KB, 8-way	8cc
LLC	2MB/core, 16-way	20cc
DRAM	4GB, tRP=tRCD=tCAS=11	variable

[1] https://github.com/ChampSim/ChampSim

Workloads

- SPEC CPU 2006 benchmark suite
- SPEC CPU 2017 benchmark suite
- Industrial workloads provided by Qualcomm for Championship Value Prediction (CVP-1)
- Big Data workloads
  - GAP benchmark suite
  - XSBench

SimPoint Methodology

TLB MPKI > 1

State-of-the-art Data TLB Prefetchers[1]

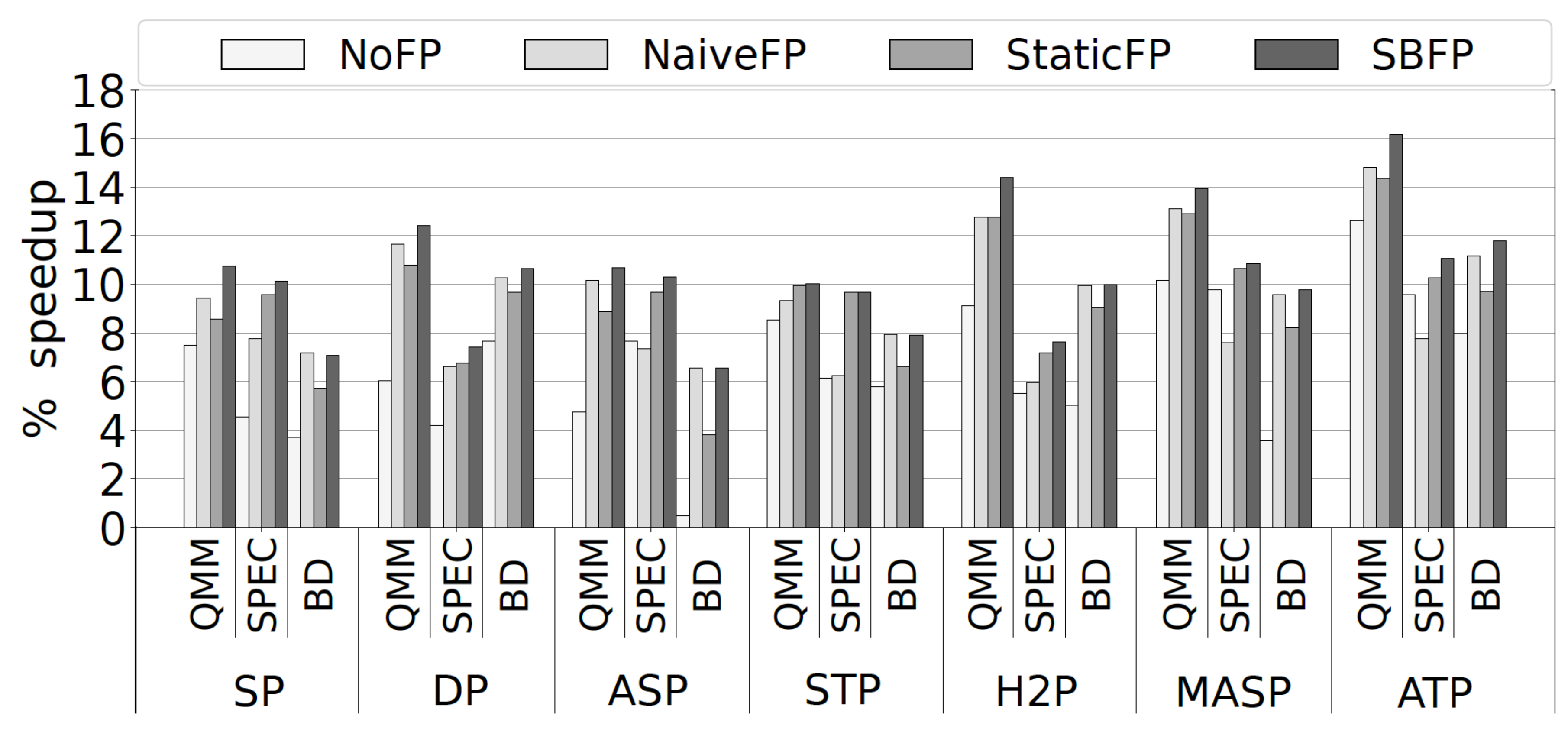
- Sequential Prefetcher (SP)
    - Prefetch the PTE of the page located next to the one produced the TLB miss
- Distance Prefetcher (DP)
    - Distances between pages that produce consecutive TLB misses
- Arbitrary Stride Prefetcher (ASP)
    - PC-correlated prefetcher

[1] G. B. Kandiraju and A. Sivasubramaniam, "Going the Distance for TLB Prefetching: An Application-driven Study", ISCA'02

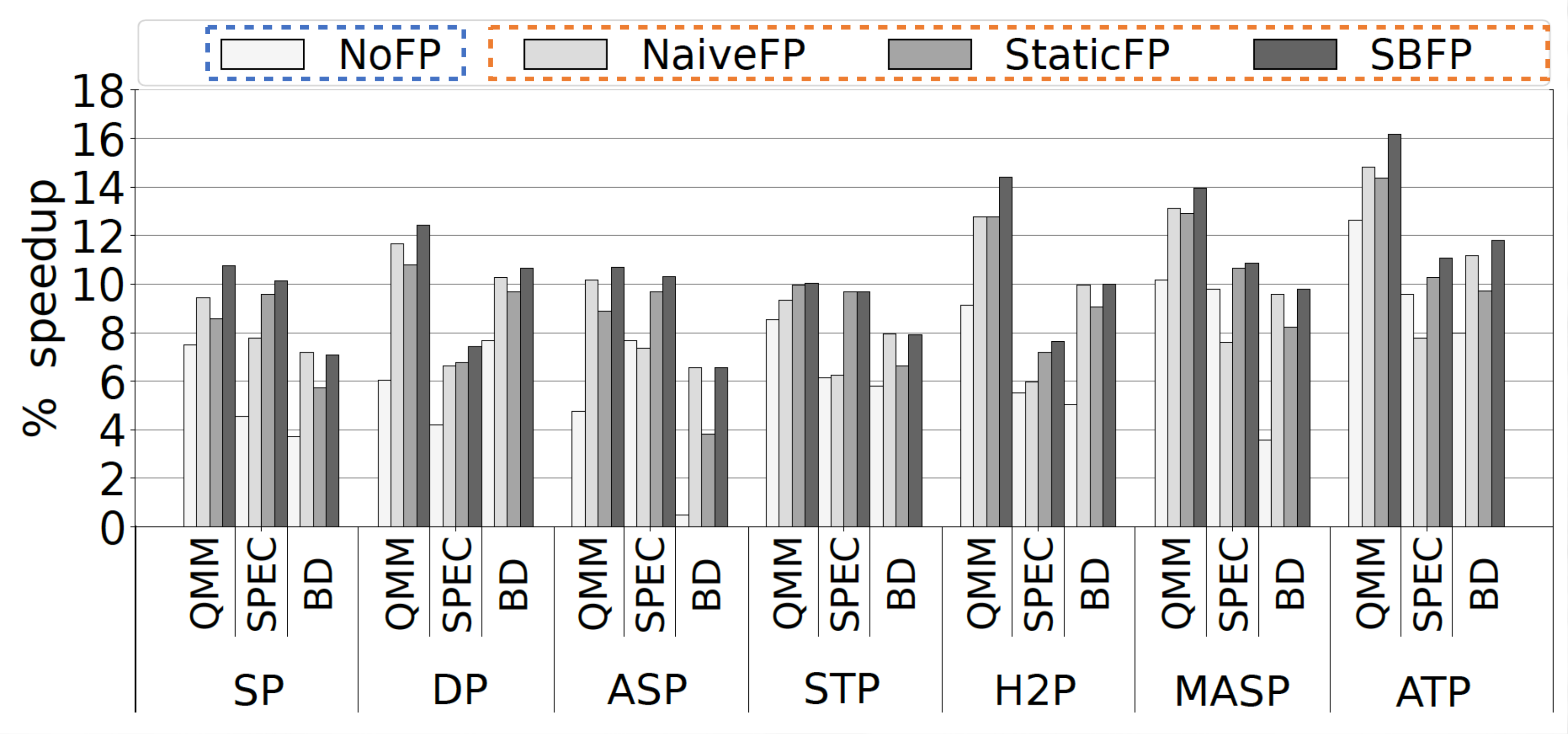
Scenarios that Exploit Free TLB Prefetching

- No Free TLB Prefetching (NoFP)
    - Free prefetching is not exploited
- Naive Free TLB Prefetching (NaiveFP)
    - All free PTEs are stored into the PQ
- Static Free TLB Prefetching (StaticFP)
    - Static offline exploration of the overall most useful free distances per TLB prefetcher

Performance Results

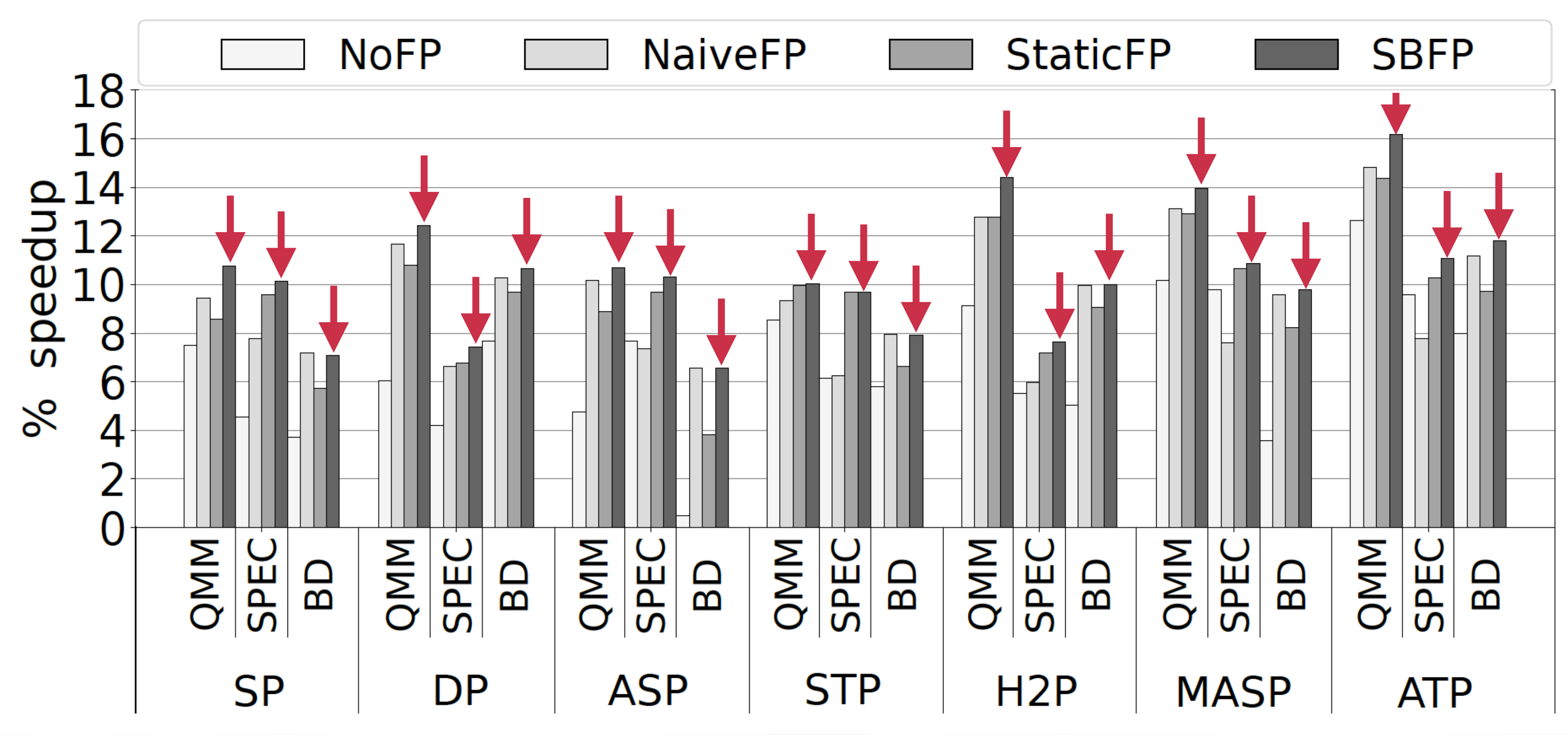


Performance Results



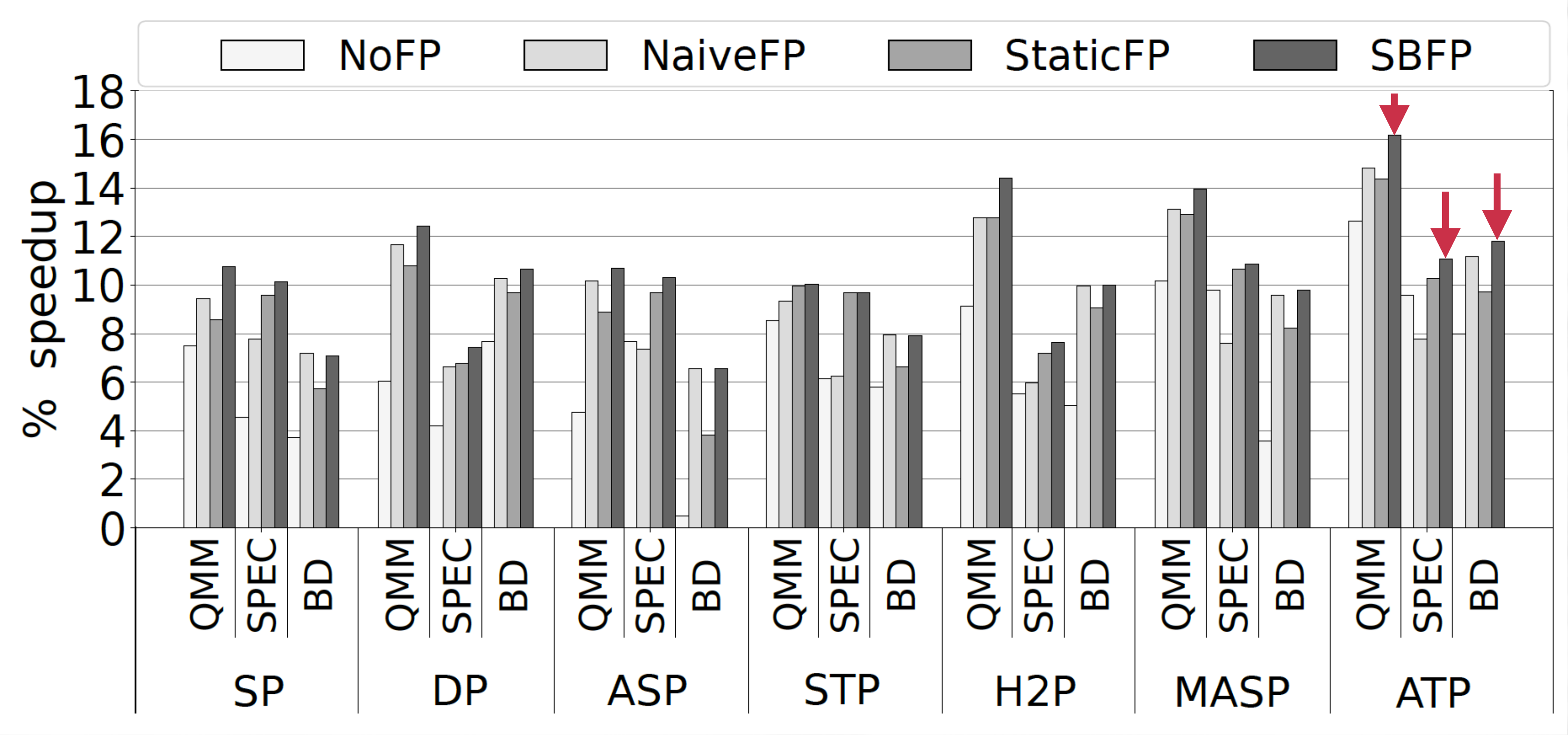
The scenarios that exploit page table locality (NaiveFP, StaticFP, SBFP) significantly improve performance over NoFP

Performance Results



SBFP provides larger speedups than the other scenarios that exploit page table locality across all evaluated TLB prefetchers

Performance Results



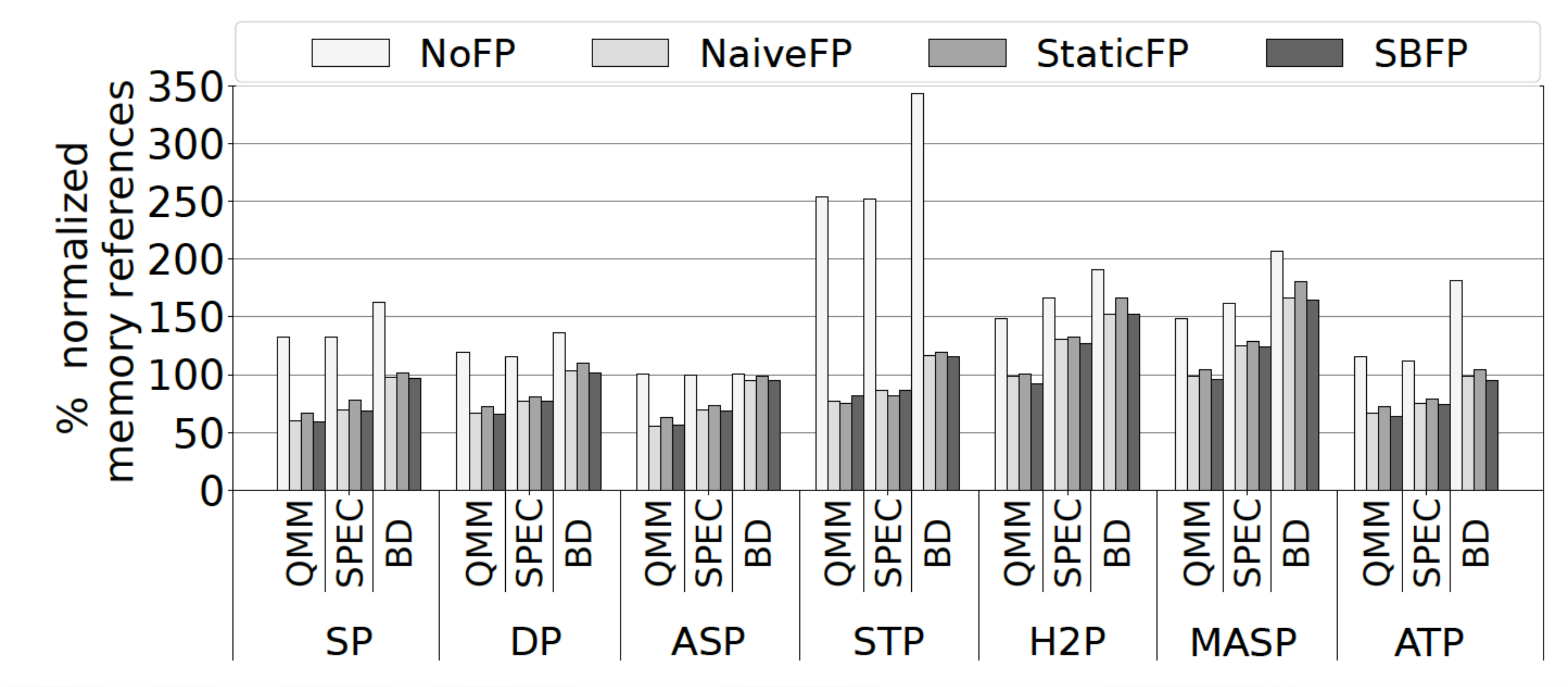
ATP combined with SBFP provides the overall best speedups

Performance Results

ATP+SBFP improves geomean performance by 16.2%, 11.1%, and 11.8% for the Qualcomm, SPEC, and Big Data workloads

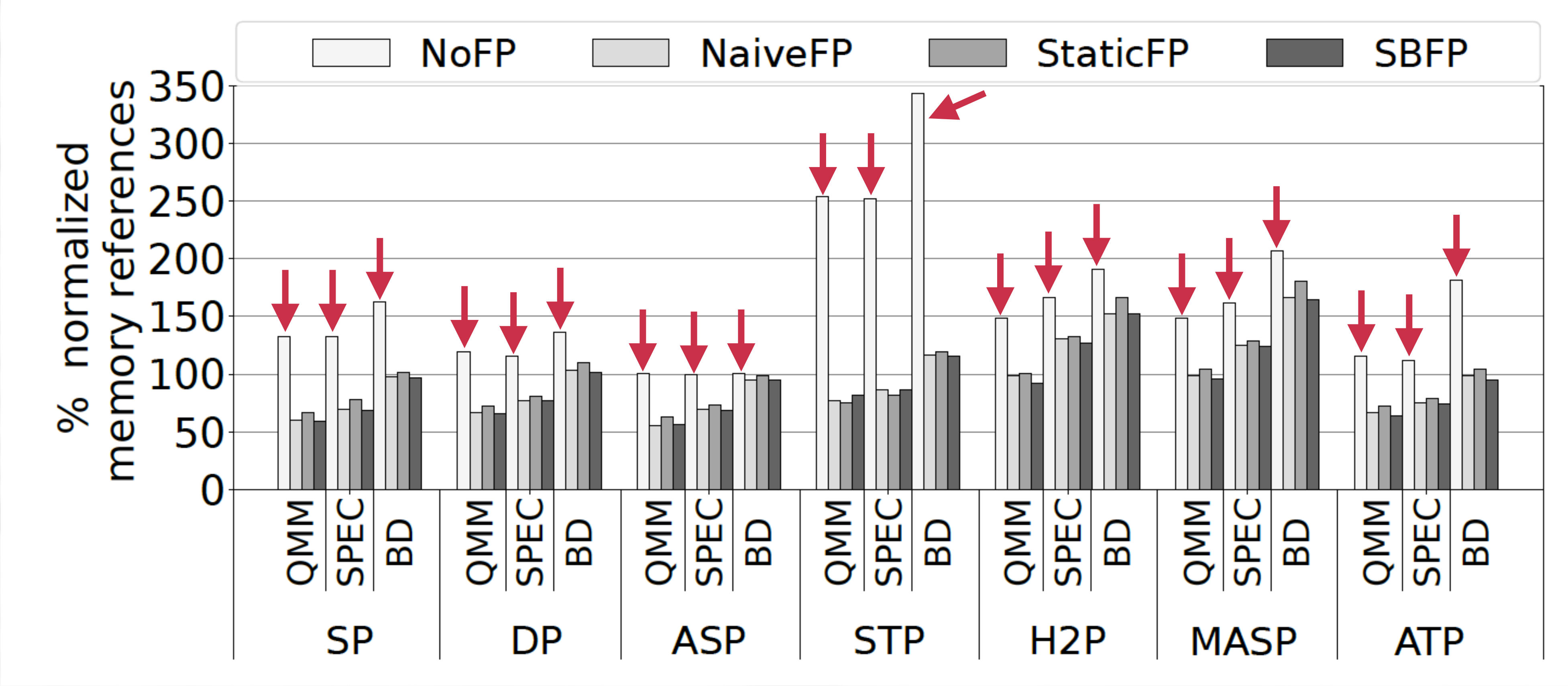
ATP+SBFP outperforms the best already proposed TLB prefetcher by 8.7%, 3.4%, and 4.2% for the Qualcomm, SPEC, and Big Data workloads

Page Walk Memory References



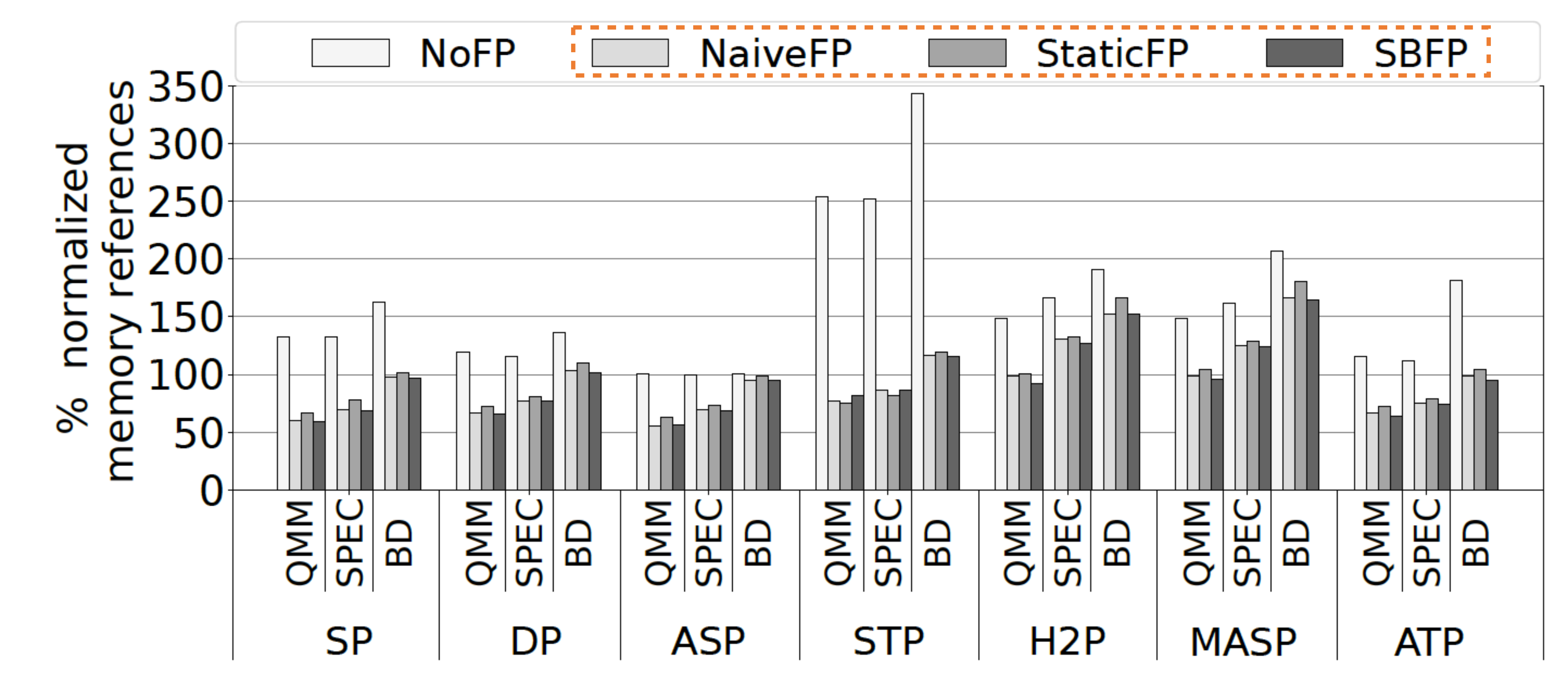
‘page walk memory reference’ is a reference to the memory hierarchy (L1, L2, LLC, DRAM) due to a page walk

Page Walk Memory References



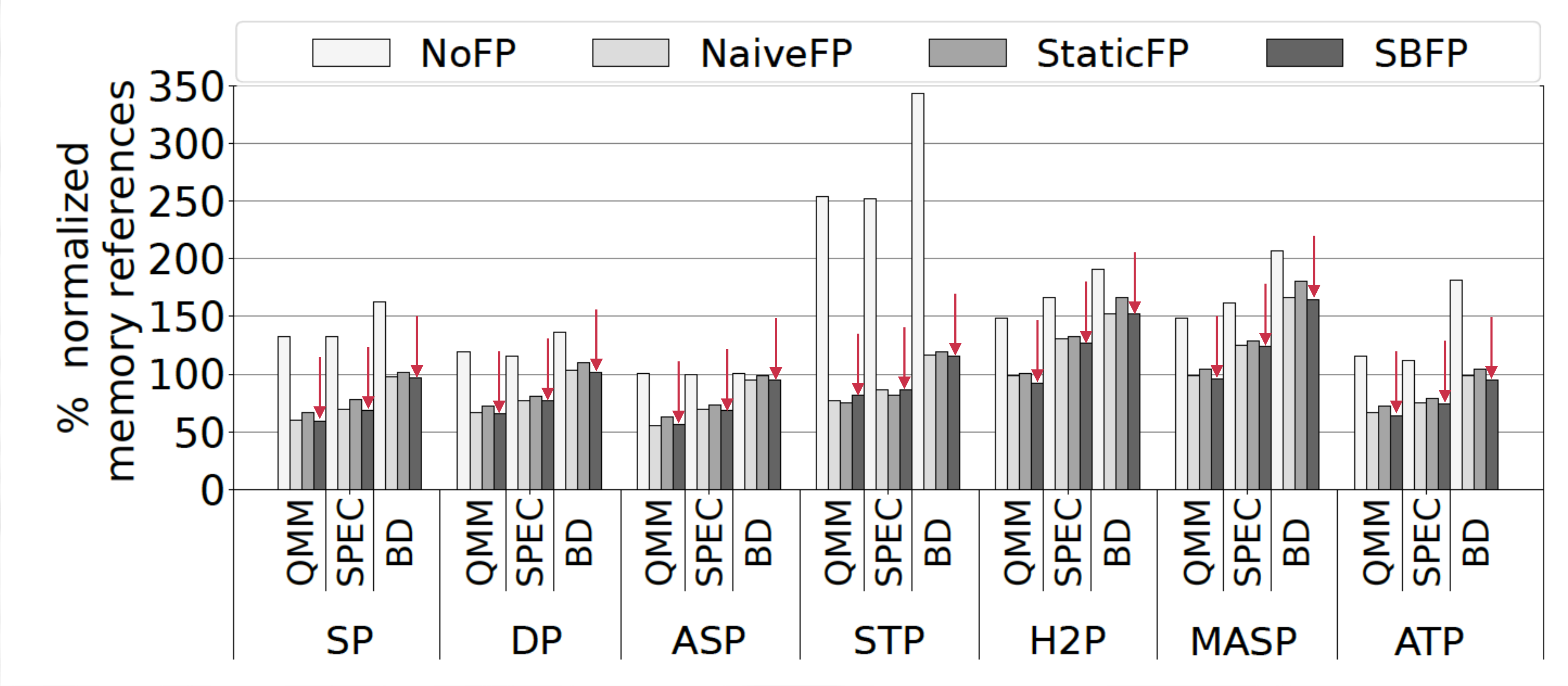
Large increase in page walk memory references when page table locality is not exploited (NoFP)

Page Walk Memory References



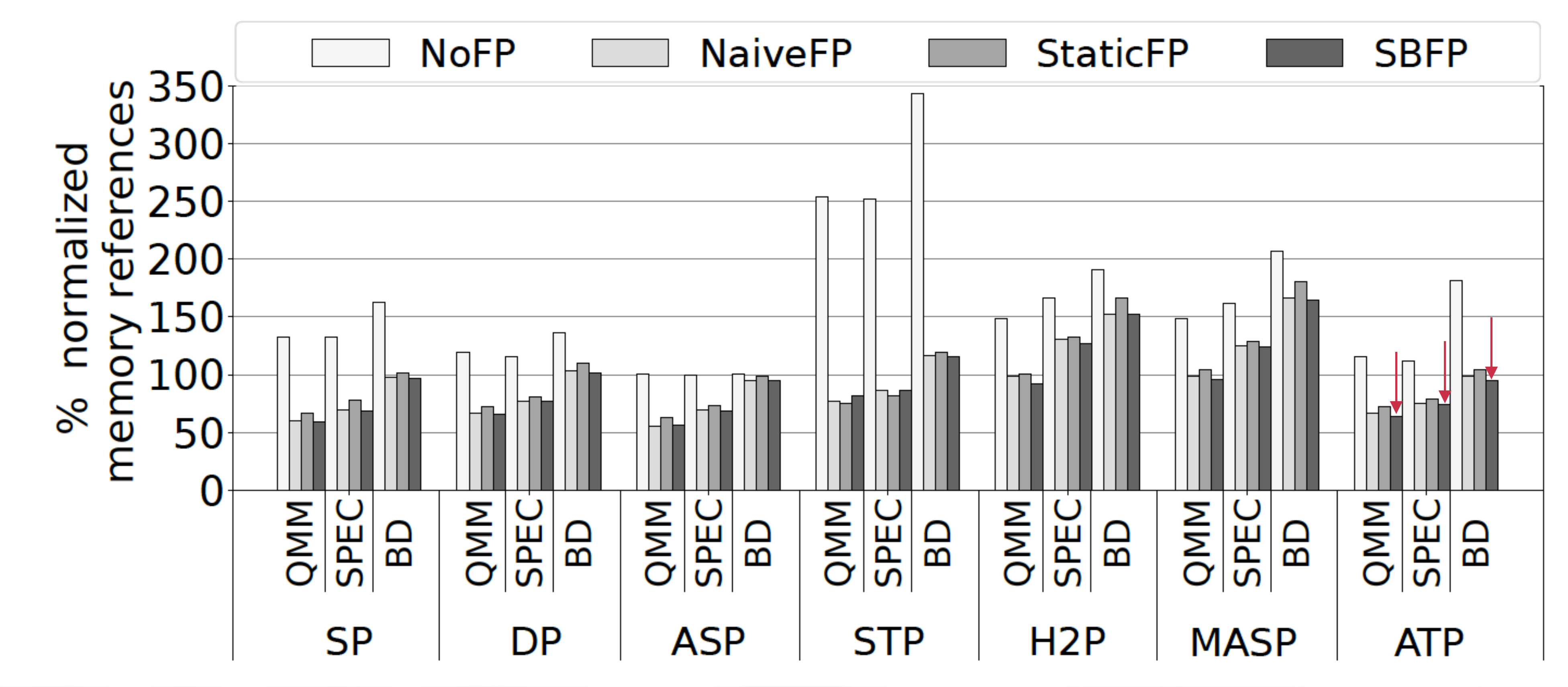
The scenarios that exploit page table locality (NaiveFP, StaticFP, SBFP) significantly reduce the page walk memory references

Page Walk Memory References



SBFP provides the highest reduction in page walk memory references across all evaluated TLB prefetchers

Page Walk Memory References



ATP combined with SBFP provides the overall highest reduction in page walk memory references

Page Walk Memory References

ATP+SBFP reduces the page walk memory references by 37%, 26%, and 5% for the Qualcomm, SPEC, and Big Data workloads, over a baseline without TLB prefetching

Storage Overhead

- SBFP requires 0.31KB of storage
- ATP requires 1.68KB of storage
- ATP combined with SBFP requires 1.99KB of storage
- ATP combined with SBFP outperforms the ISO-storage[1] scenario by more than 9% across all the benchmark suites

[1] L2-TLB is augmented with additional entries to match the storage budget of ATP+SBFP

Additional Evaluation Results

- Activation ratio of the constituent prefetchers of ATP
- Portion of the PQ hits provided by ATP and SBFP
- Performance improvement when 2MB pages are used
- Dynamic energy consumption of address translation
- Comparison with other approaches that improve TLB performance

Conclusions

- This work reduces the address translation overheads via TLB prefetching
  - microarchitectural technique | relies on the memory access patterns | not disruptive
- Our proposal
  - Sampling-based Free TLB Prefetching (SBFP)
    - Exploit page table locality to enhance the performance of prior and novel TLB prefetchers
  - Agile TLB Prefetcher (ATP)
    - Composite TLB prefetcher that combines three low-cost TLB prefetchers and disables prefetching when the TLB miss stream is irregular

Combining ATP with SBFP improves geomean performance by more than 10% across different benchmark suites and reduces most of the page walk references to the memory hierarchy

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

GOBIERNO
DE ESPAÑA

MINISTERIO
DE CIENCIA, INNOVACIÓN
Y UNIVERSIDADES

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Generalitat
de Catalunya

Thank you!

georgios.vavouliotis@bsc.es