Exercise Session 3

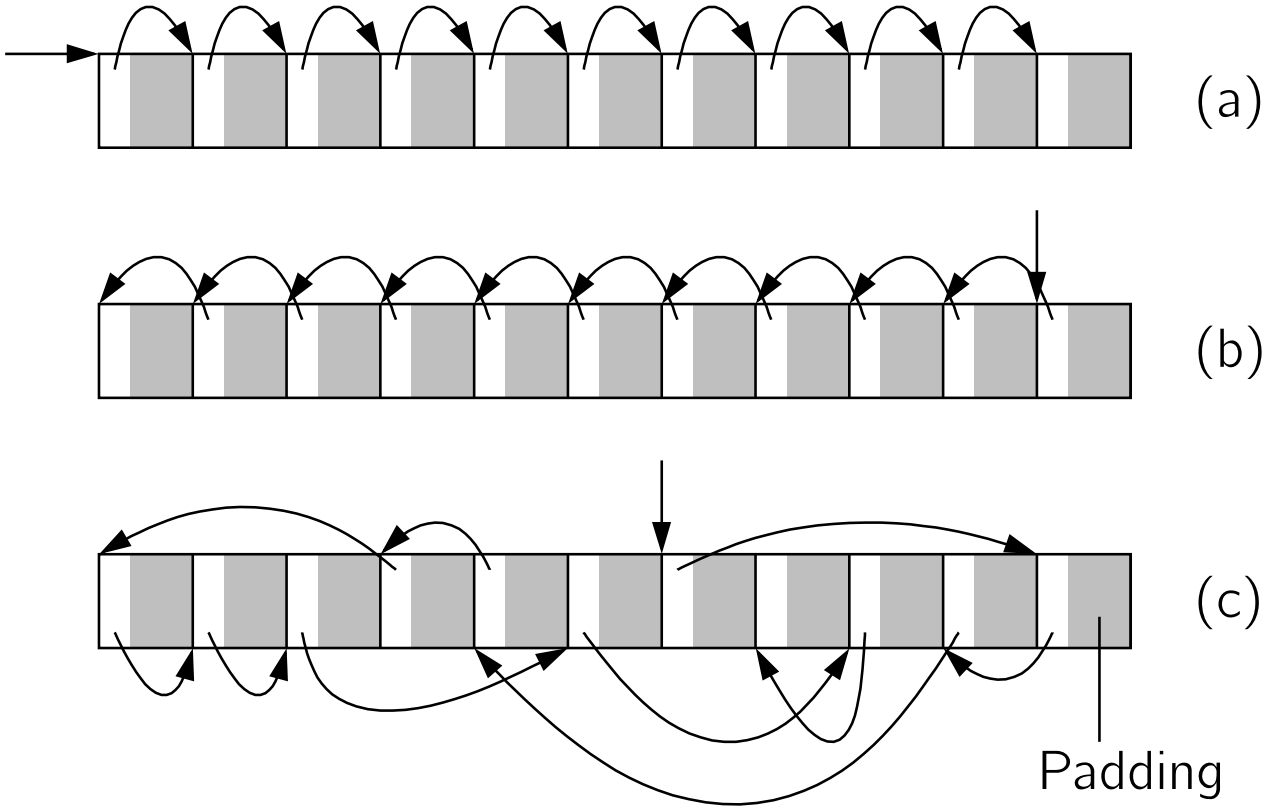
Data Processing on Modern Hardware
263-3502-00L — Fall Semester 2012

Cagri Balkesen
cagri.balkesen@inf.ethz.ch

Department of Computer Science ETH Zurich, Switzerland

4 October 2012

Memory Traversal using Linked Lists



(a): increasing order, (b): decreasing order, (c): random order

Task & Evaluation Platforms

Traverse list multiple times and measure wall clock time

- Vary size of list
- Vary size of elements
- Vary access patterns

Linked List with padding array:

```
struct listelement {
    struct listelement *next;
    long int padding[NUMPAD];
}
```

Evaluation Platforms

1. Intel, Core Architecture (2006)
2. Intel, Nehalem Architecture (2009)
3. (SGI), MIPS Architecture (1996)
4. Sun, SPARC Architecture (1998)

Tricking the Compiler (1)

- Prevent the compiler from "optimizing" loops

```
#define ONE     head = head->next;
#define FIVE    ONE ONE ONE ONE ONE
#define TEN     FIVE FIVE
#define FIFTY   TEN TEN TEN TEN TEN
#define HUNDRED FIFTY FIFTY

getcyclecount(&start);
for (t=0; t<numtries; t++) {
  HUNDRED;
}
getcyclecount(&end);
```

Tricking the Compiler (2)

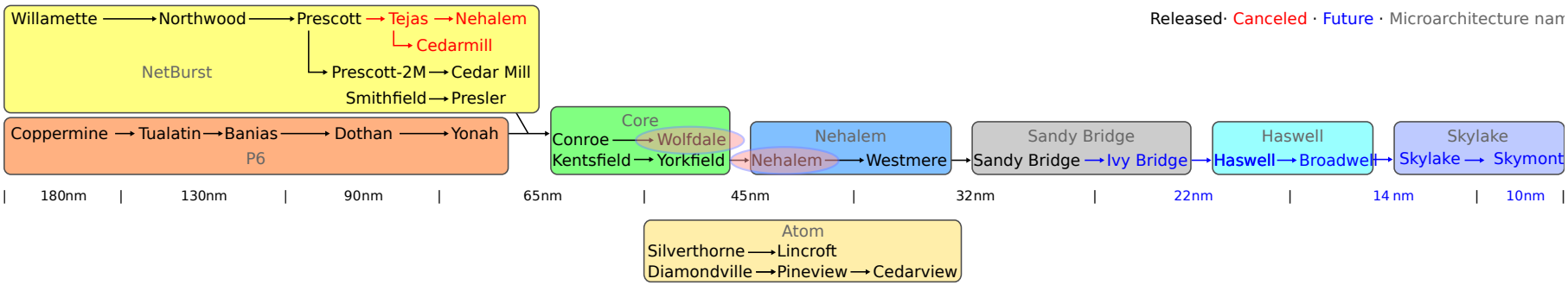
- Prevent the compiler from removing "dead code"

```
getcyclecount(&start);
for (t=0; t<numtries; t++) {
 HUNDRED;
}
getcyclecount(&end);

/* prevent dead code elimination of heap */
__asm __volatile(
 "orq %0, %0 \n\t"
 : : "r"(head)
 );
```

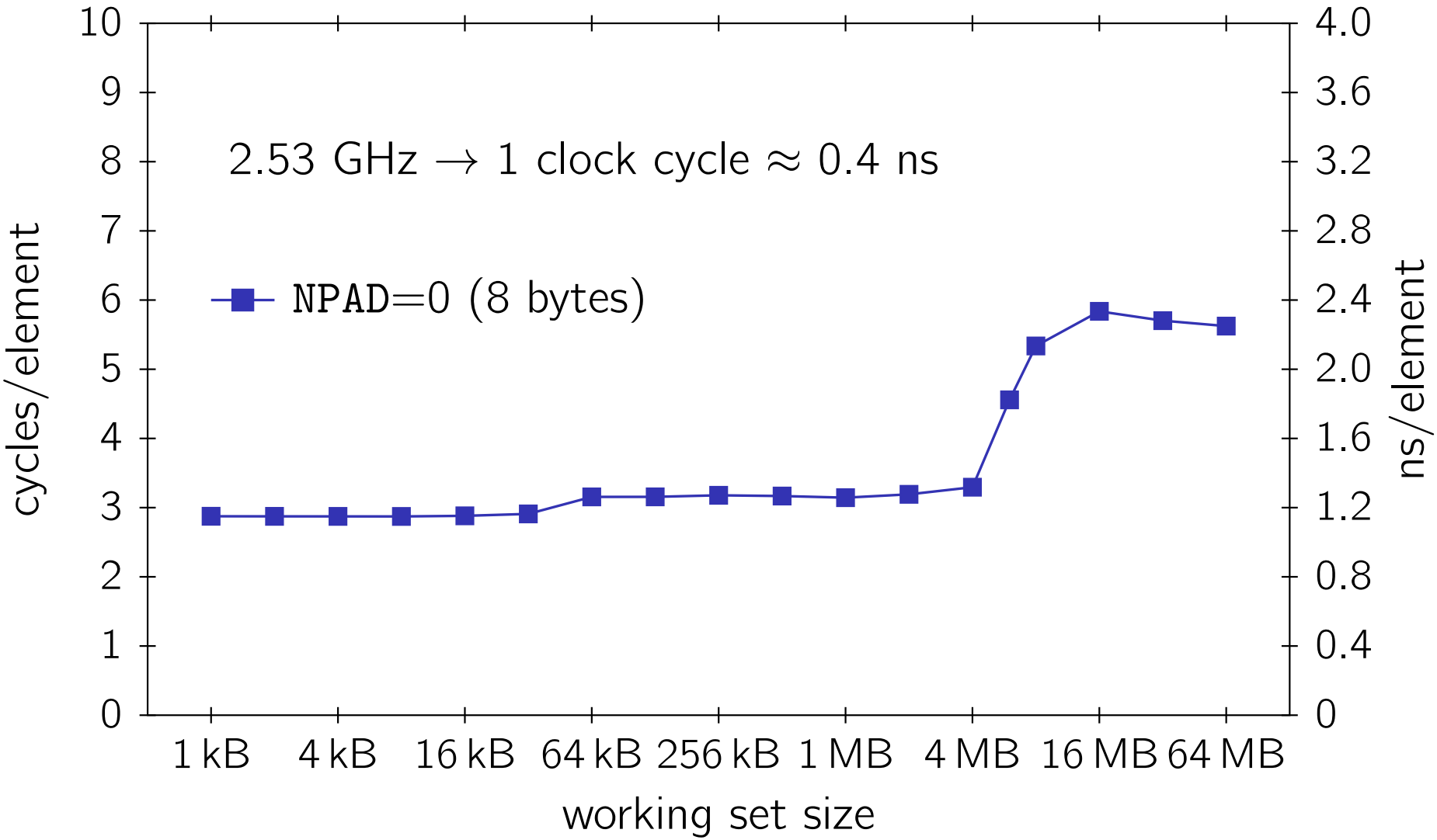
Intel Tick-Tock Model

- Every "tick" is a shrinking of process technology of the previous microarchitecture
- Every "tock" is a new microarchitecture

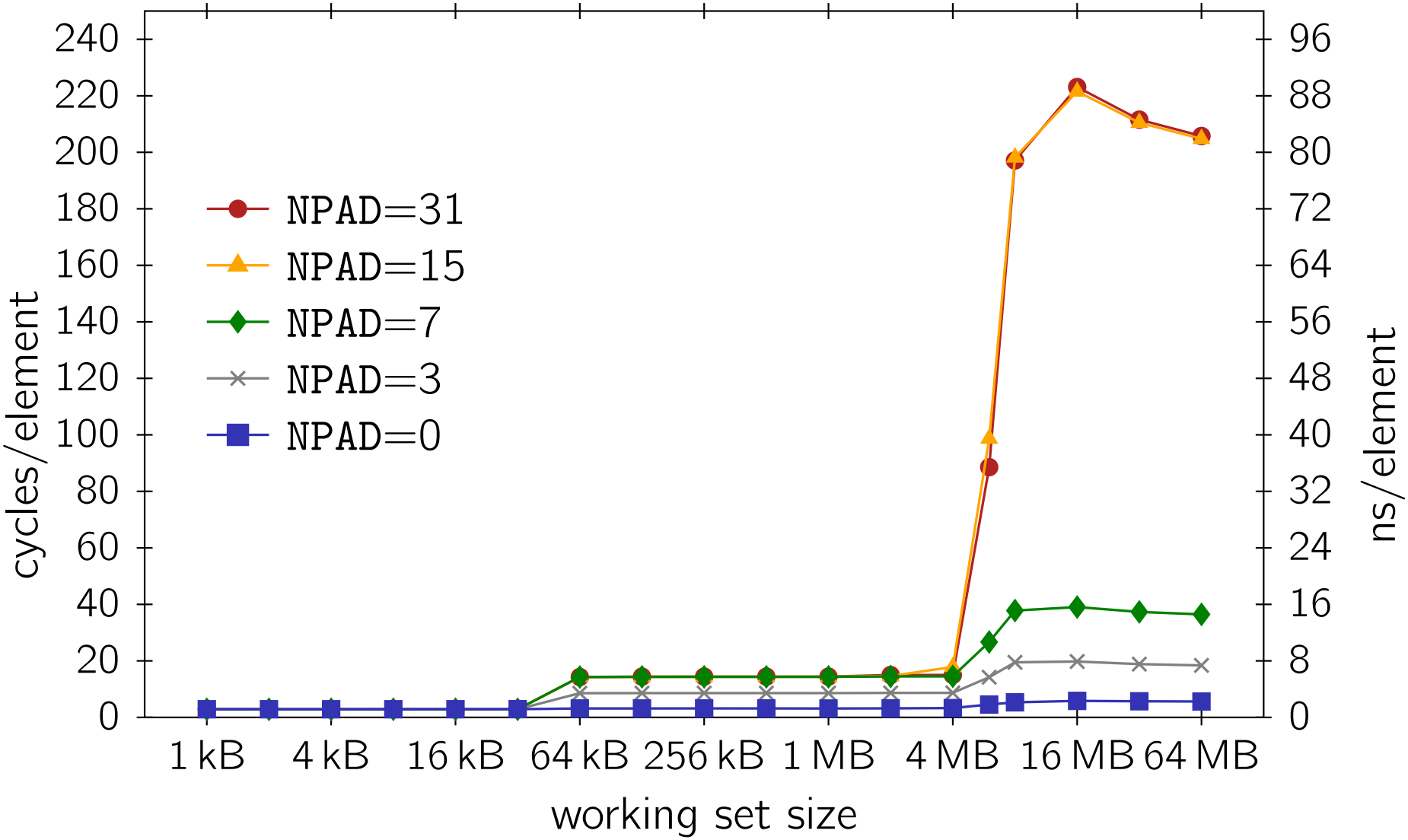


- Source: http://en.wikipedia.org

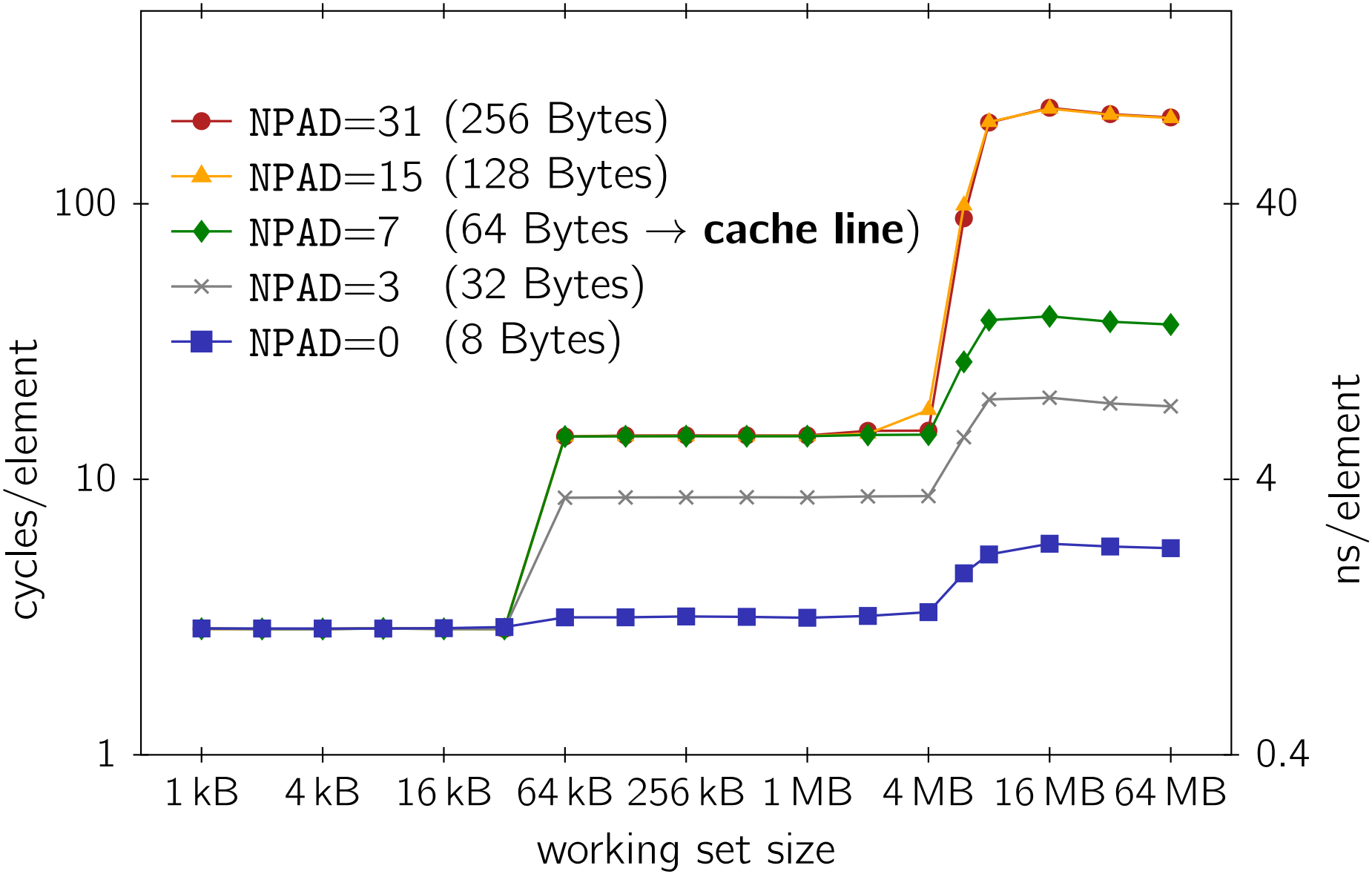
Core 2 Duo (Penryn): Sequential Access: NPAD=0



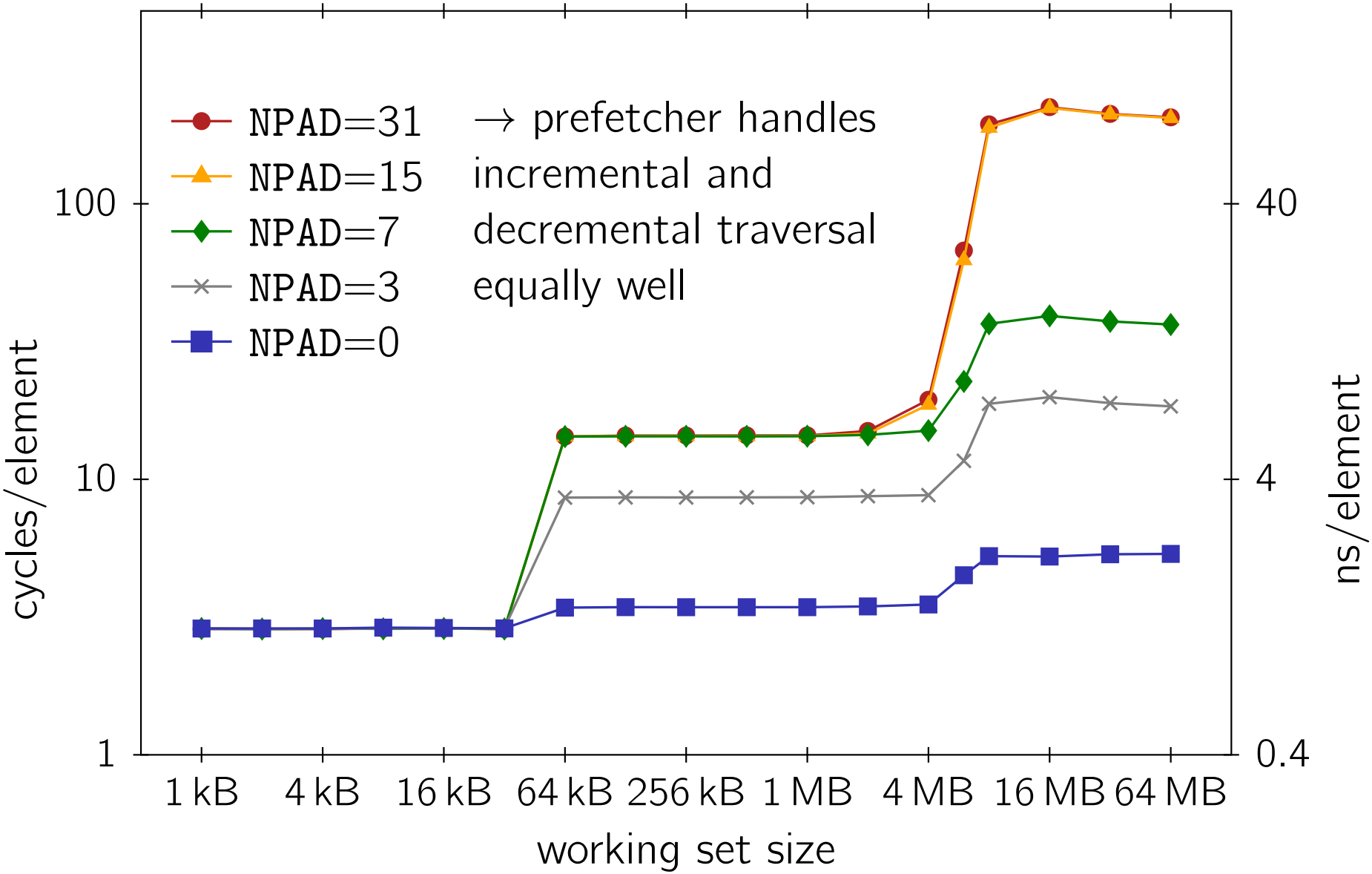
Core 2 Duo (Penryn): Varying Stide Width



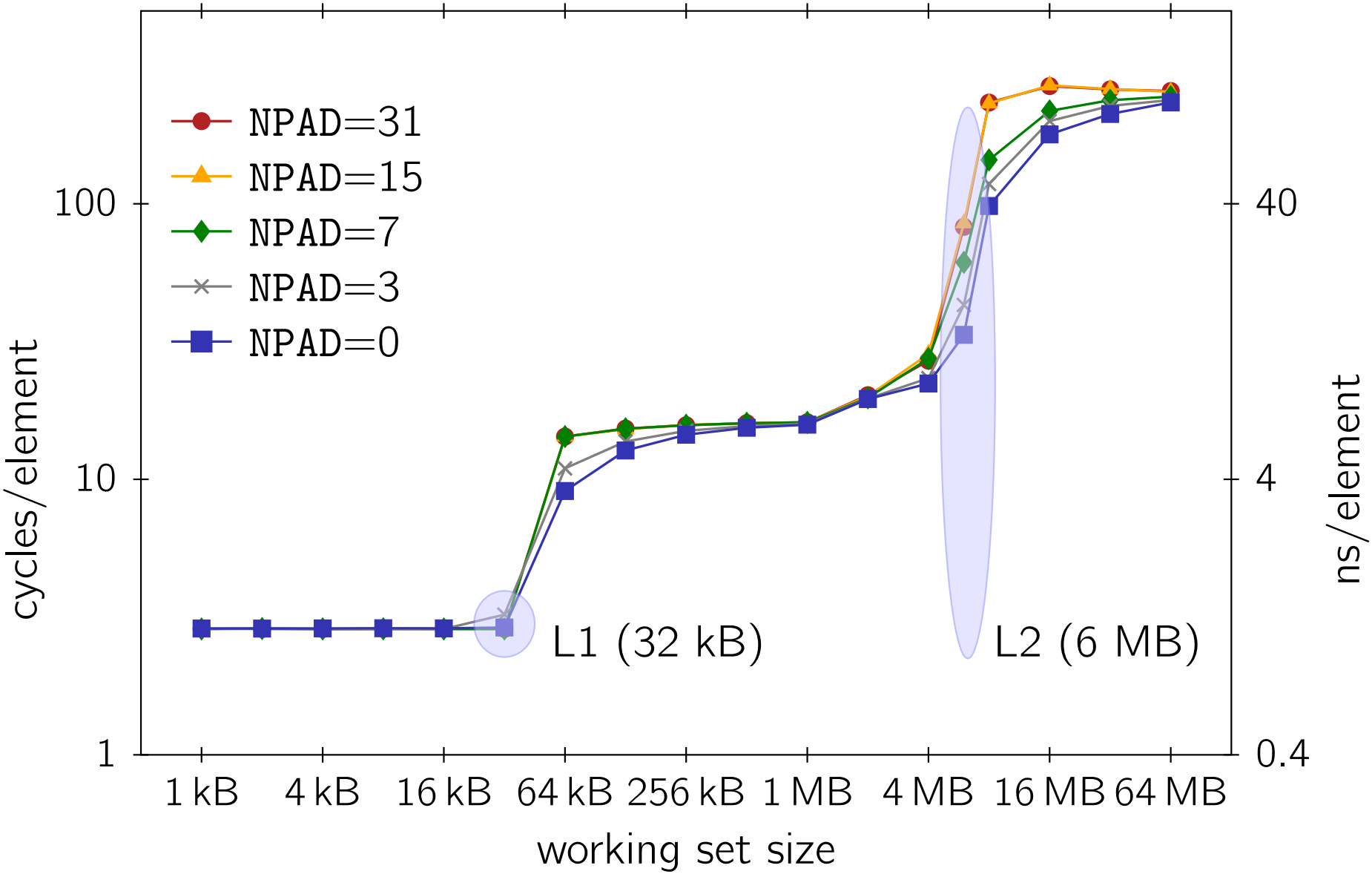
Core 2 Duo (Penryn): Varying Stide Width (Log Scale)



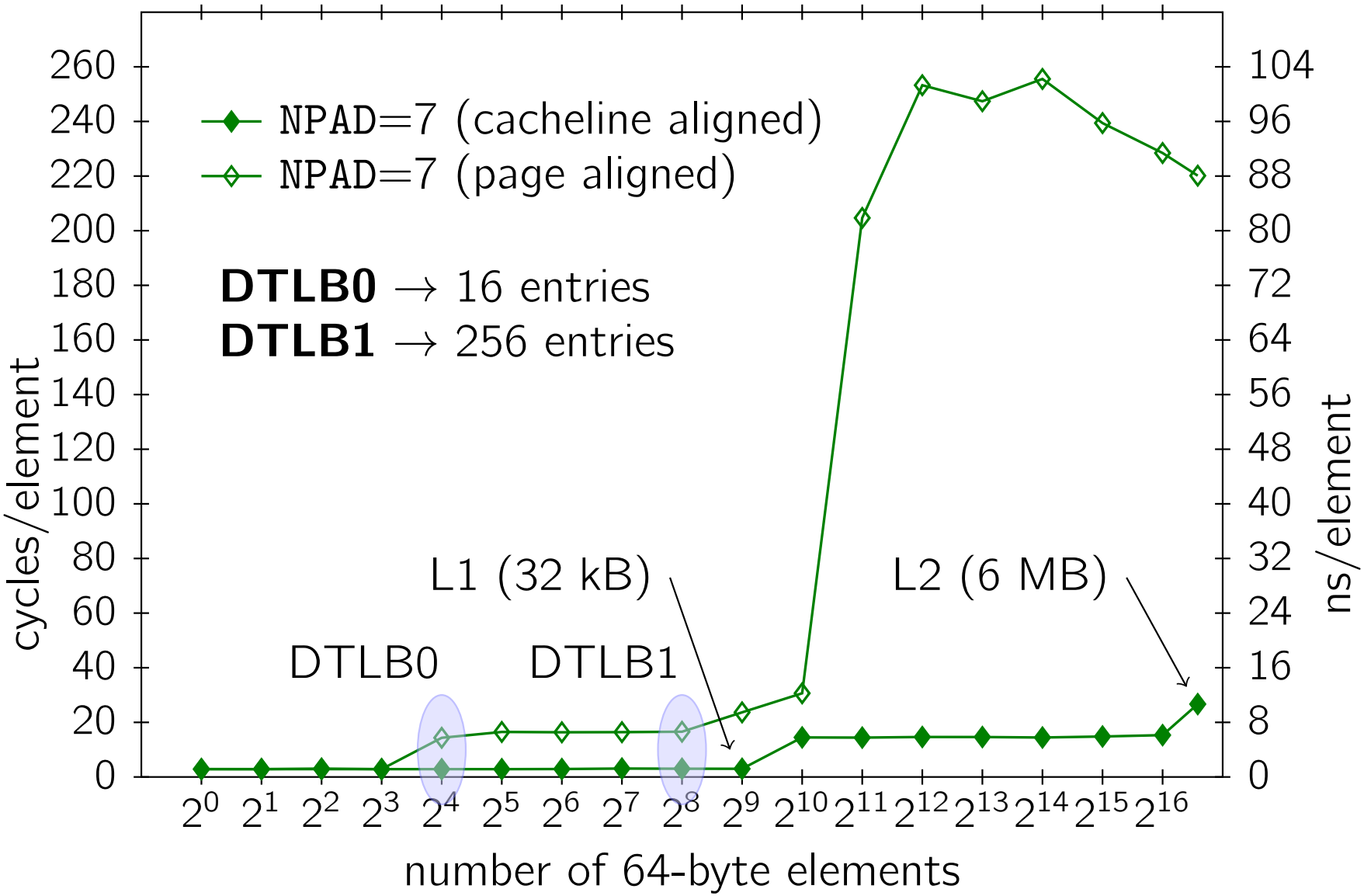
Core 2 Duo (Penryn): Backward Traversal



Core 2 Duo (Penryn): Random Access



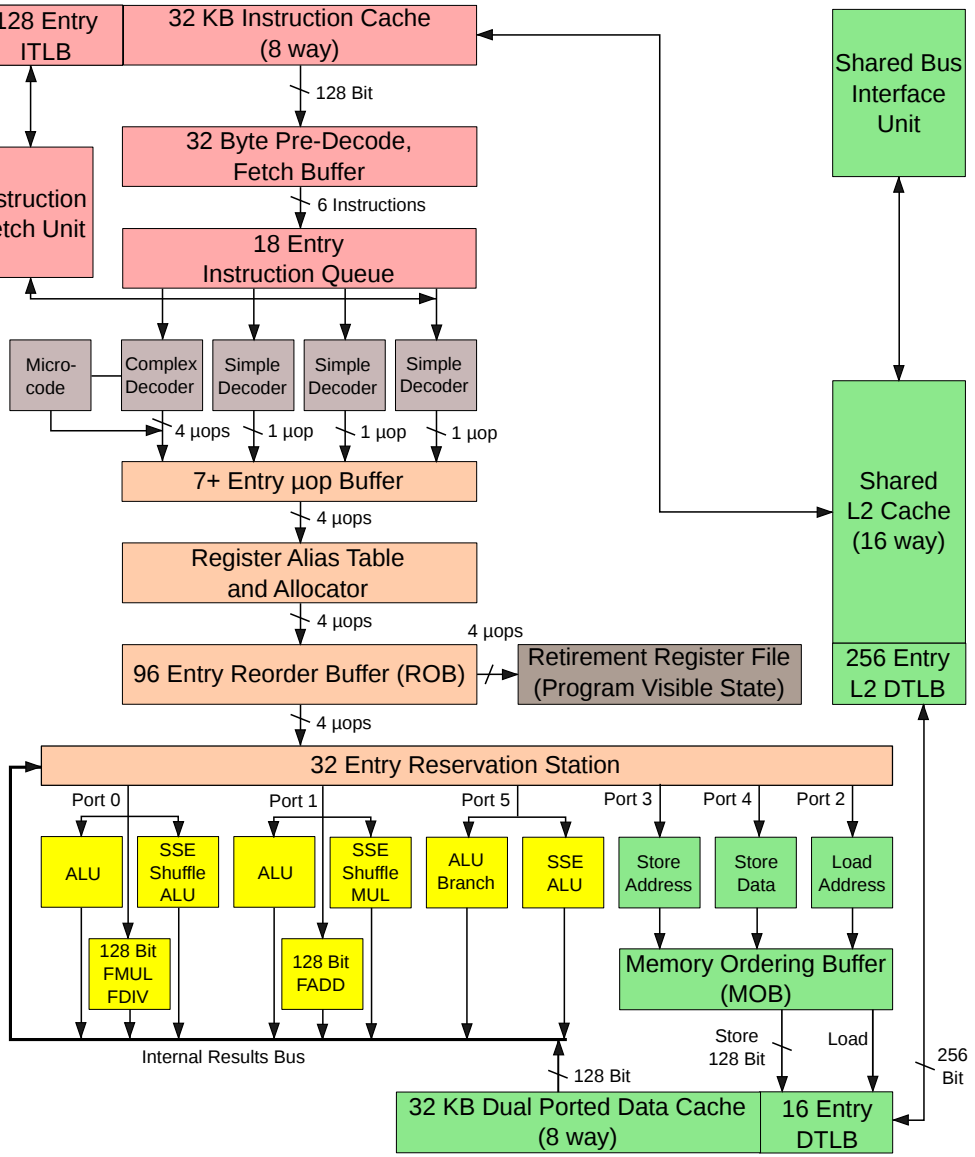
Core 2 Duo (Penryn): TLB Experiment (NPAD=7)



Core 2 Duo (Penryn): Summary

- General
  - Model name : Intel(R) Core(TM)2 Duo CPU P9600
  - 2 cores/die @2.53GHz
  - Microarchitecture : Penryn (mobile variant of Wolfsdale)
  - Process : 45nm
- Cache
  - L1 instruction : **32 kB** (8-way)
  - L1 data : **32 kB** (8-way)
  - L2 unified shared :**6MB** (24-way)
- TLB
  - DTLB0: **16** (4-way) entries (4KB page)
  - ITLB0: **128** (4-way) entries (4KB page)
  - DTLB1: **256** (4-way) entries (4KB page)

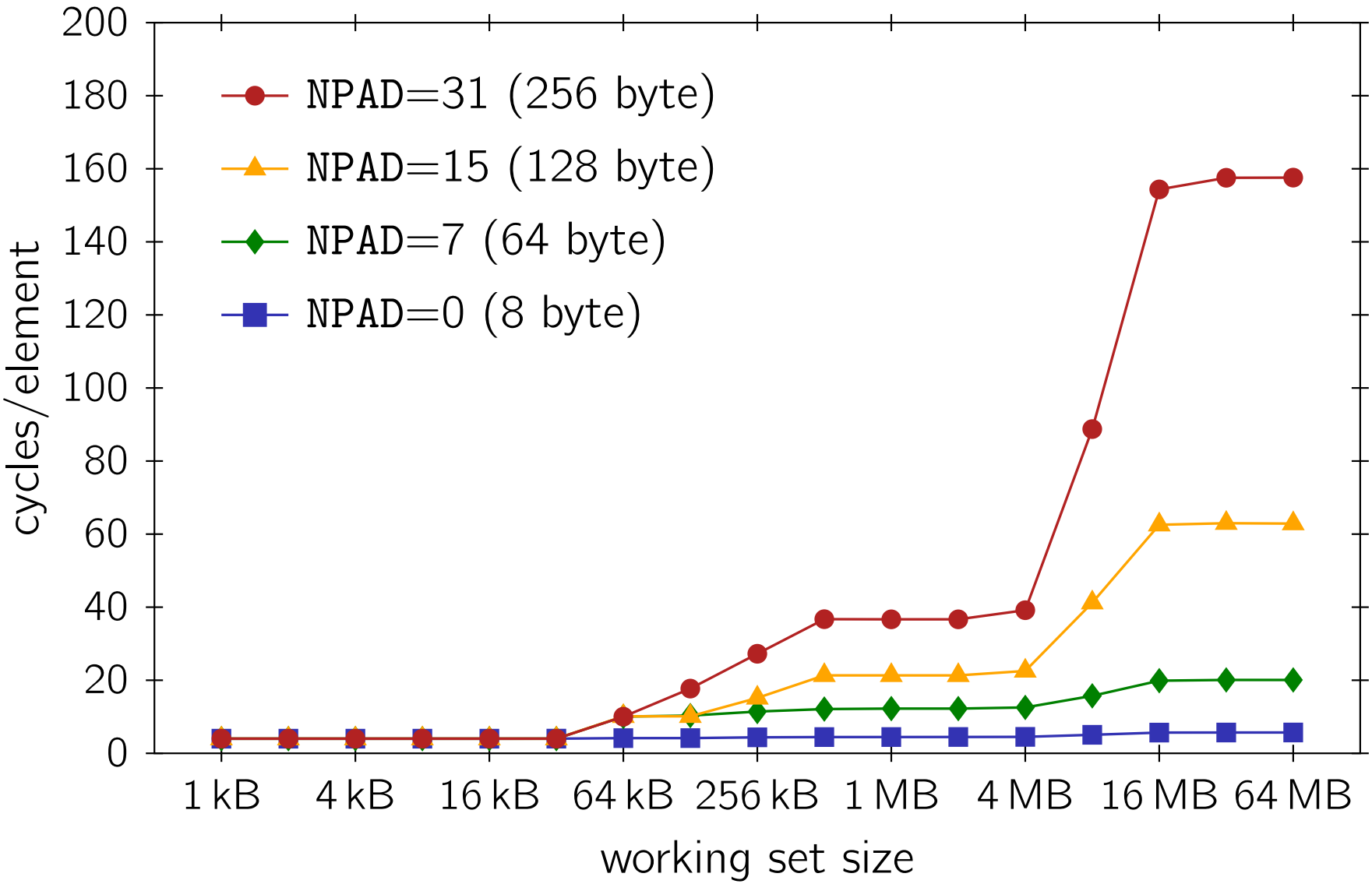
Intel Core Architecture



Intel Core 2 Architecture

source: http://en.wikipedia.org

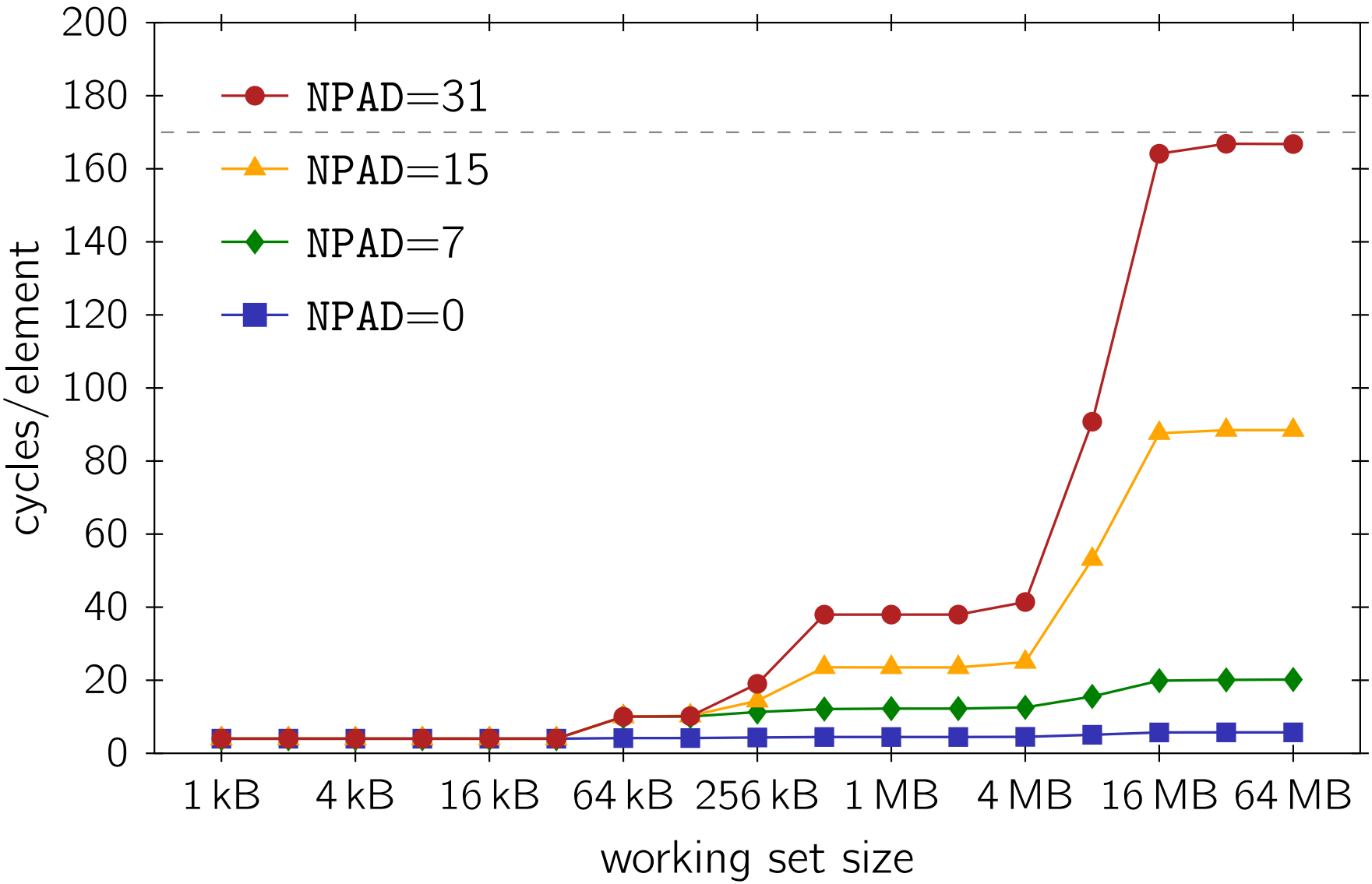
Nehalem: Sequential Access — Forward Traversal



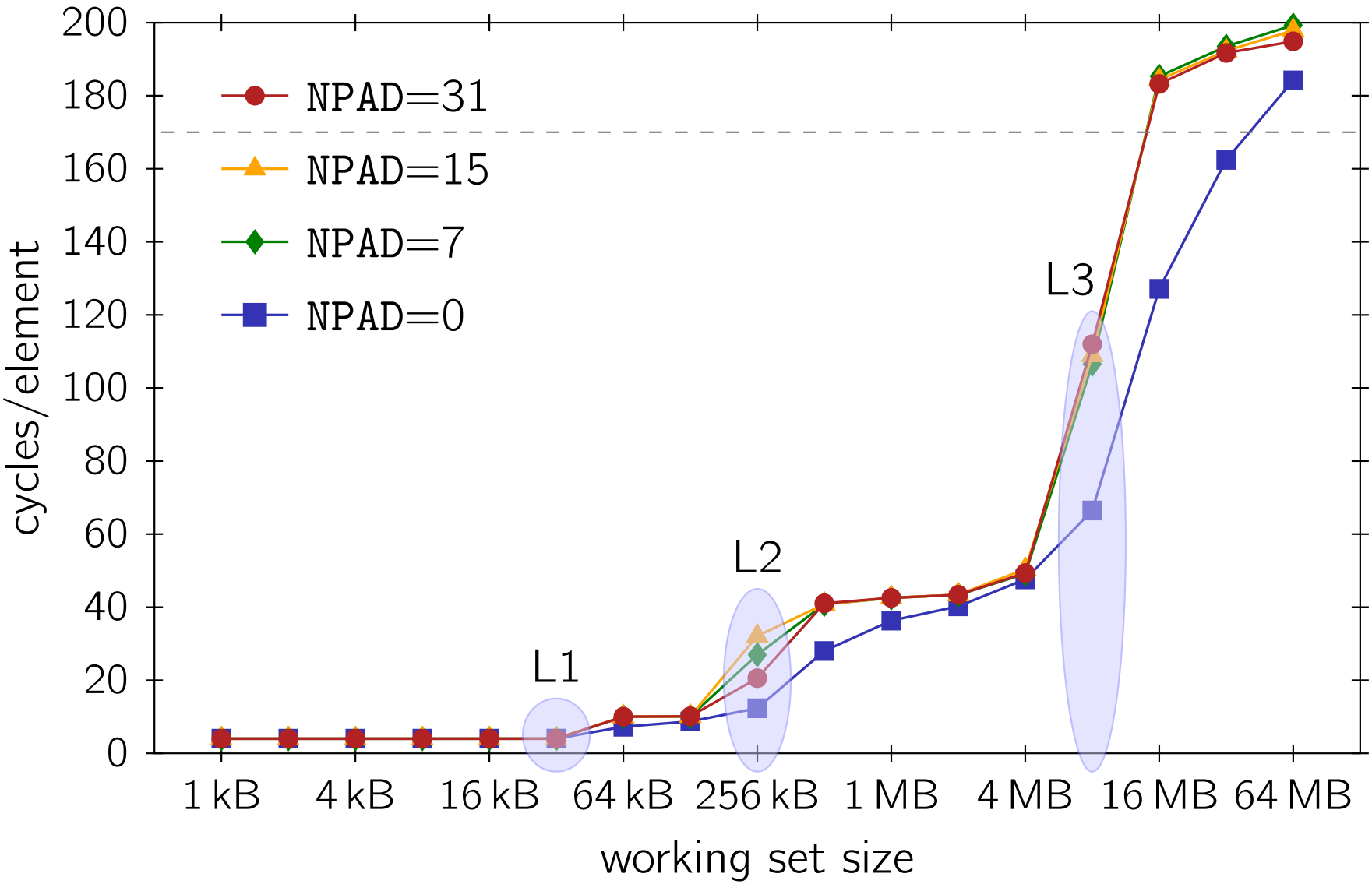
Nehalem: Prefetching Unit

- **Data Cache Unit (DCU) Prefetcher (Streaming Prefetcher)**
  - Prefetches next cache line into L1D if multiple loads from same cache line within a fixed period of time
- **Instruction Pointer-based Strided Prefetcher (IPSP)**
  - Prefetcher keeps sequential load history information per instruction pointer
  - Prefetch into L1D triggered when load instructions have a regular stride (forward or backward)
- **Data Prefetch Logic (DPL)**
  - Prefetches data into L2
  - Remembers past request patterns of L1 to L2 data cache
  - Can also detect more complicated data accesses, e.g., when intermediate data blocks are skipped
  - Adjusts its prefetching effort based on the utilization of the memory to cache paths

Nehalem: Sequential Access — Backward Traversal

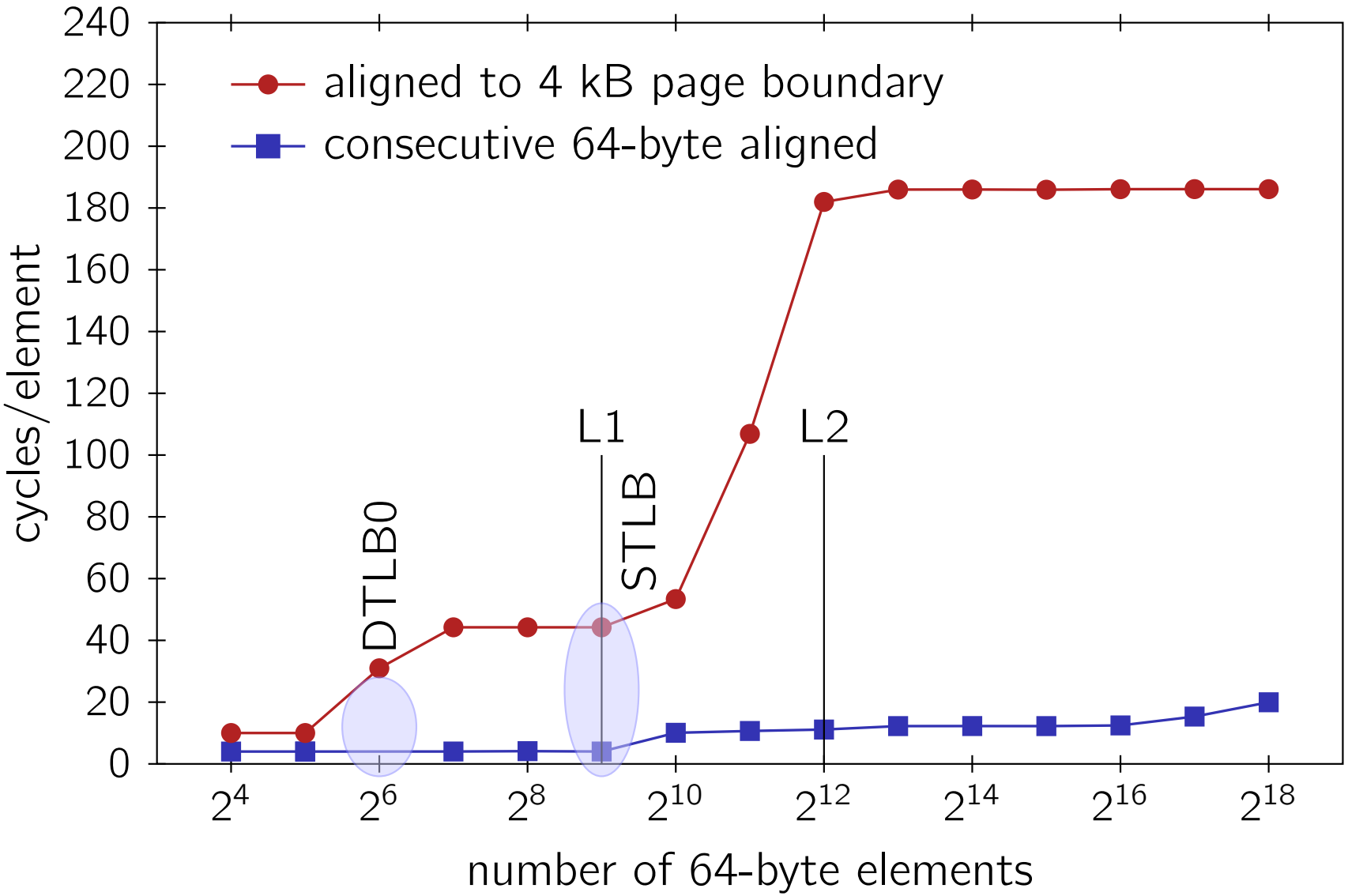


Nehalem: Random Access



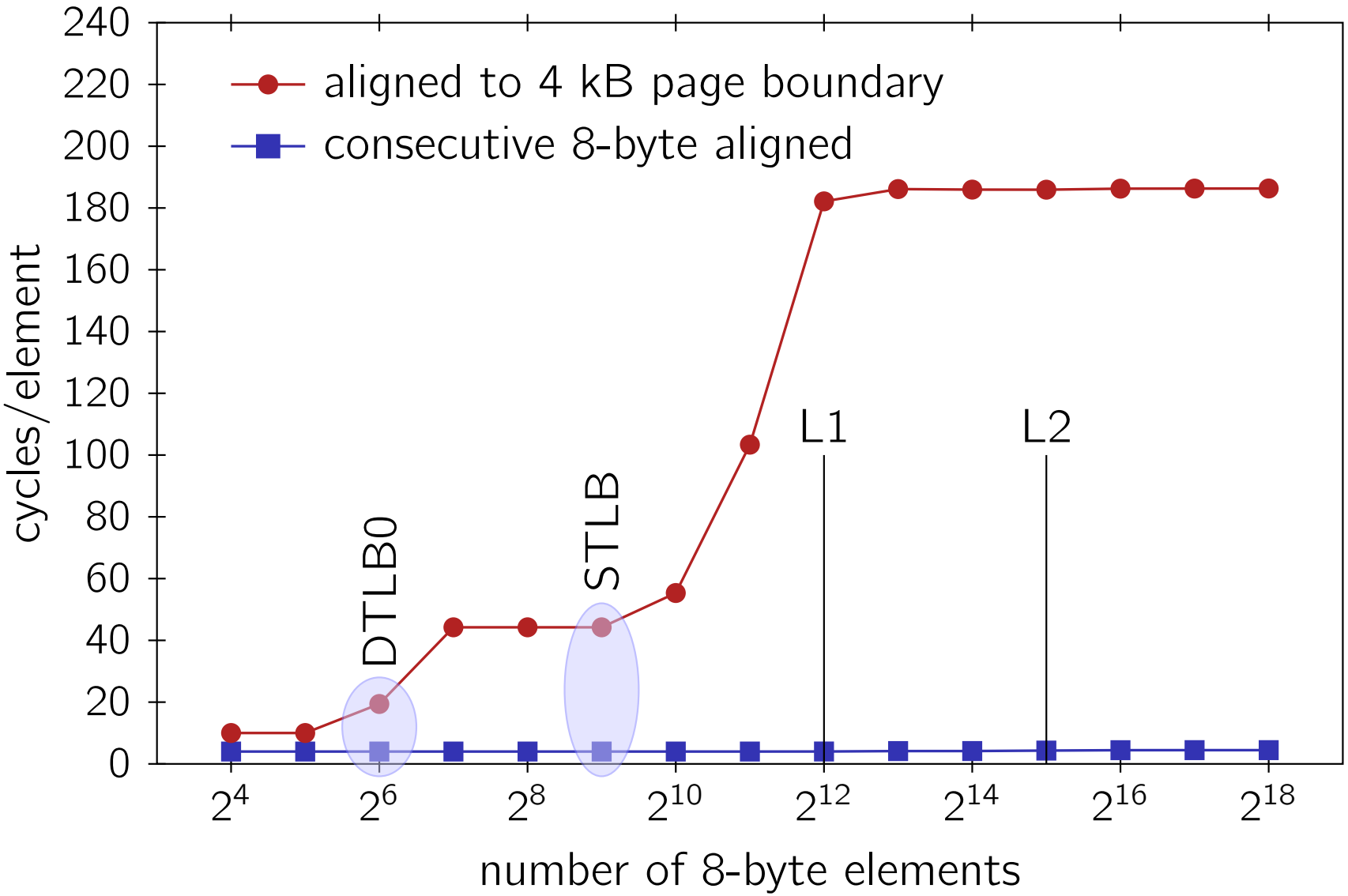
Nehalem: TLB Experiment — NPAD=7

Forward traversal with 64 bytes/element (`NPAD=7`)



Nehalem: TLB Experiment — NPAD=0

Forward traversal with 8 bytes/element (`NPAD=0`)



Nehalem : Summary (Level 1 Cache)

- Gainestown (Nehalem-EP) 4 cores/die @2260 MHz
- L1 (4x, one per core)
  - instruction: **32 kB**, 4-way set assoc., stores μOps
  - data: **32 kB**, 8-way set assoc., 64 bytes/line
  - access latency: 4 cycles, throughput: 1 element/clock
  - write update policy: writeback
- On Linux goto (or consult datasheet of CPU)
  - /sys/devices/system/cpu/cpu*/cache/index*
  - level = 1, type = Data, size = 32K
  - coherency_line_size = 64
  - number_of_sets = 64
  - ways_of_associativity = 8
  - 64(bytes)$\times$64 $\times$ 8 = 32768

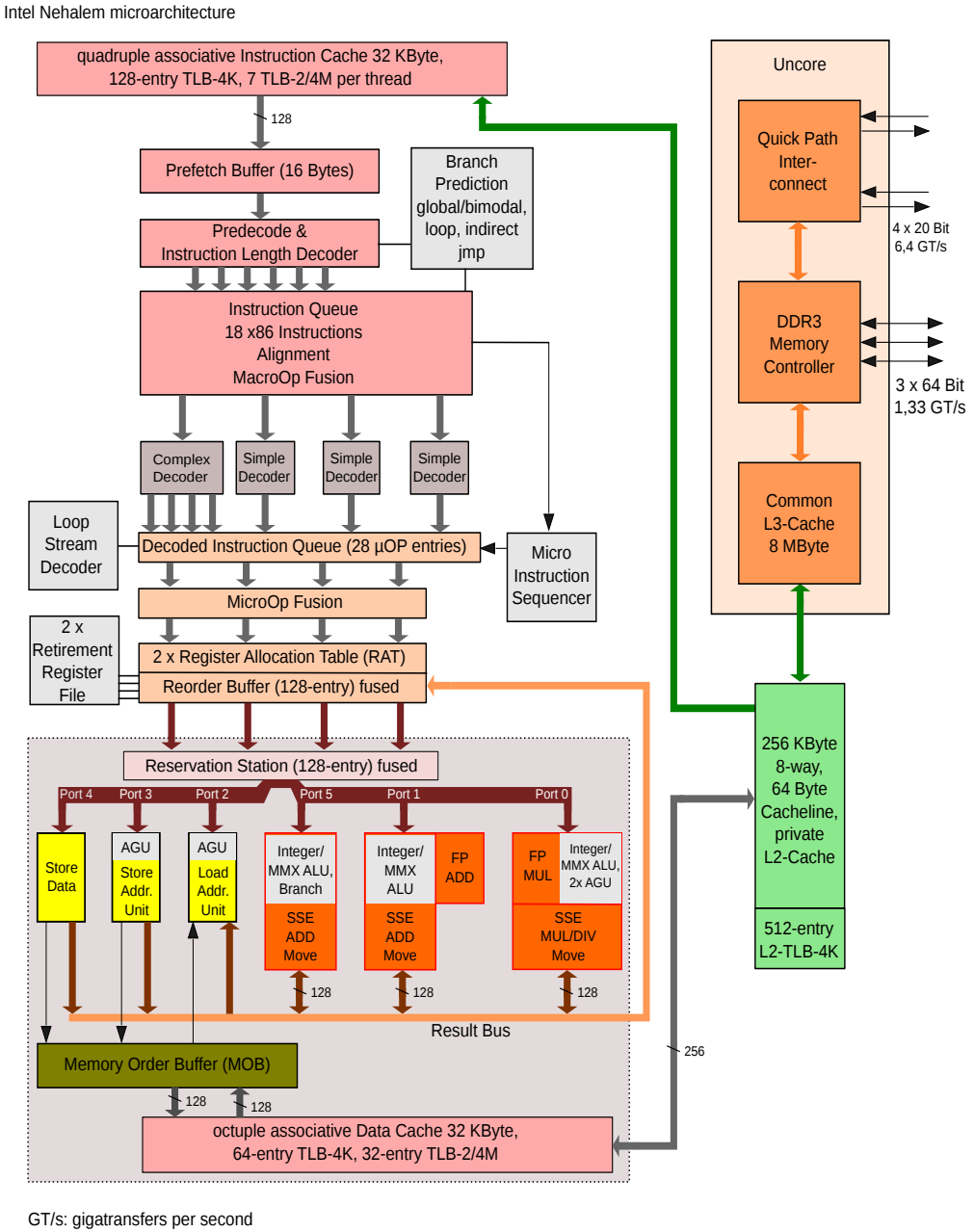
Nehalem : Summary (Level 2 & 3 Cache)

- L2 (4x, one per core)
  - unified = same for data & instructions
  - **256 kB**, 8-way set assoc., 64 bytes/line
  - L2 data non-inclusive in L1
  - access latency: 10 cycles, throughput: 1 element/clock
  - write update policy: writeback
- L3 (shared among 4 cores)
  - **8 MB**, 16-way set assoc., 64 bytes/line
  - inclusive, data in L1/L2 also in L3
  - access latency: 35–40 cycles, throughput varies, writeback

Nehalem : Summary (TLB)

- Two-level Hierarchy
- Level 1 (seperate for code+data)
  - Data DTLB0: 64 entries
  - Code ITLB: 64 entries per thread $\rightarrow$ 128
- Level 2 (unified code+data)
  - STLB: 512 entries
  - 4-way associative

Intel Nehalem Architecture

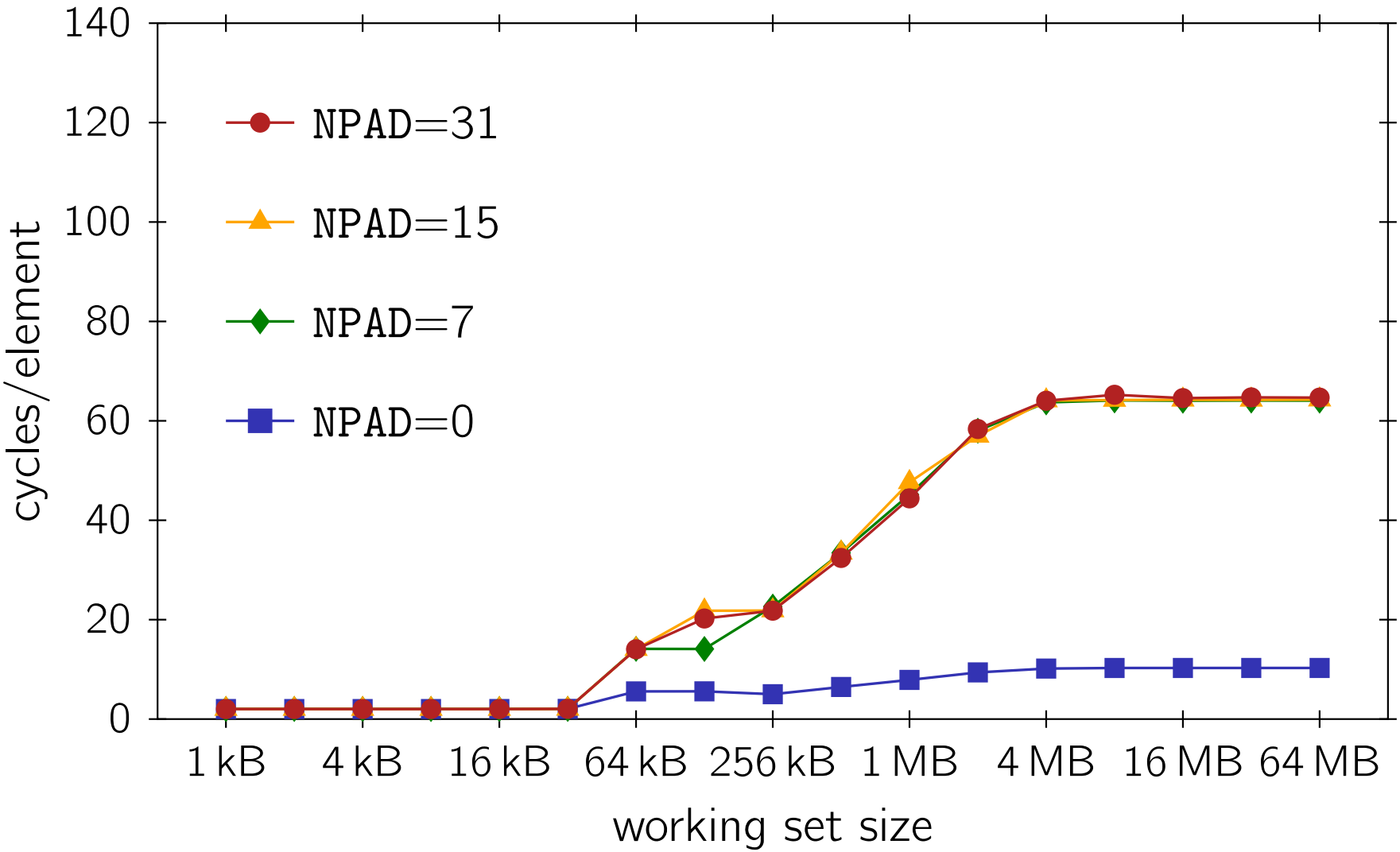


source: http://en.wikipedia.org

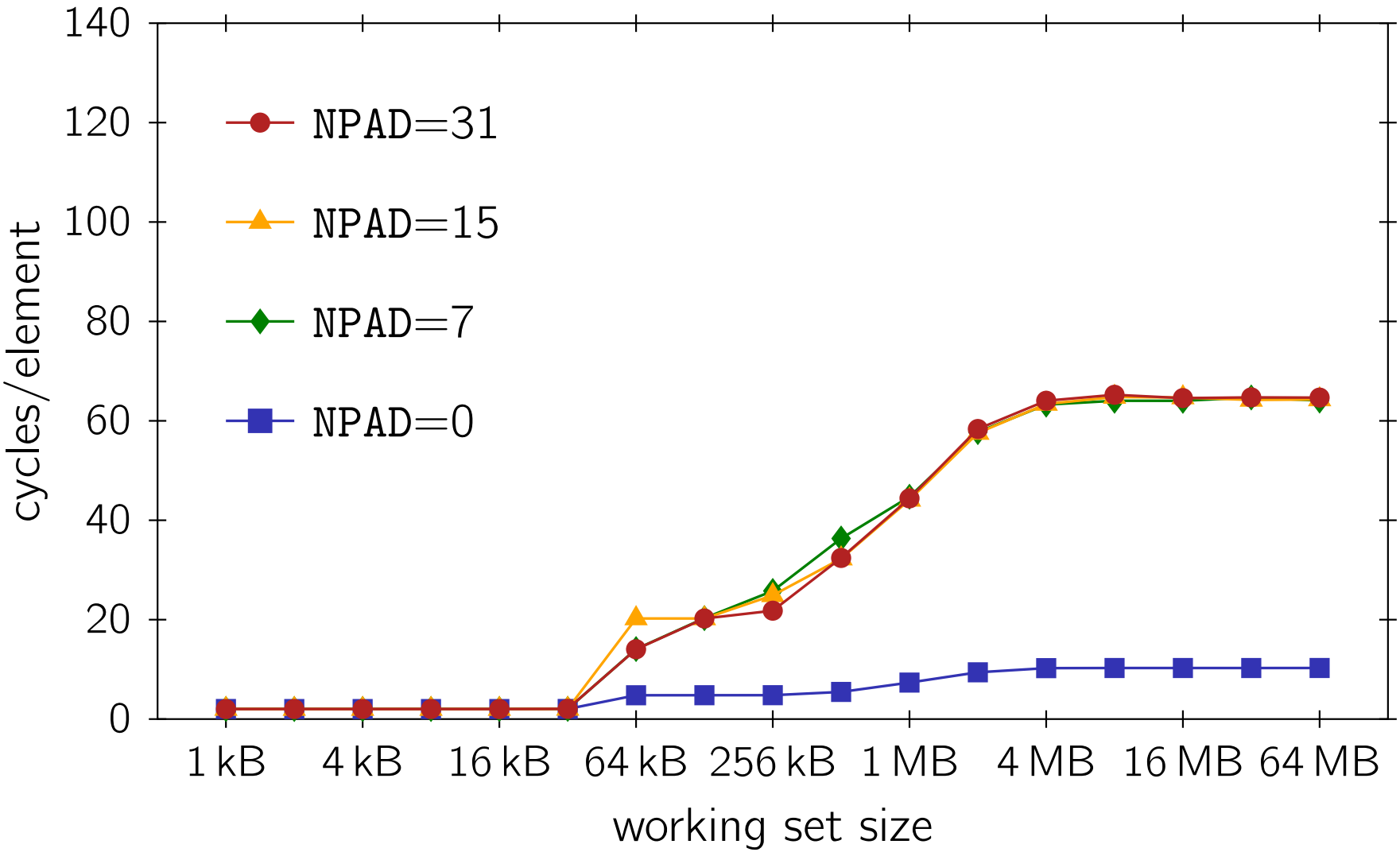
Nehalem vs. MIPS

	Nehalem L5520	**MIPS R5000**
Number of cores	4	1
CPU clock	2260 MHz	200 MHz
L1 instr.	32 kB (4-way)	32 kB (2-way)
L1 data	32 kB (8-way)	32 kB (2-way)
L2	256 kB (4-way) on-chip	1024 kB off-chip, 100 MHz
L3	8 MB (16-way)	—
TLB	ITLB: 64 entries DTLB0: 64 entries STLB: 512 entries	48 dual-entry (even/odd) $\approx$ 96 entries
TDP	60 W	10 W
Transistors	731 million	3.6 or 5.0 million
Process	45 nm	0.35 μm
Introduction	2009	1996

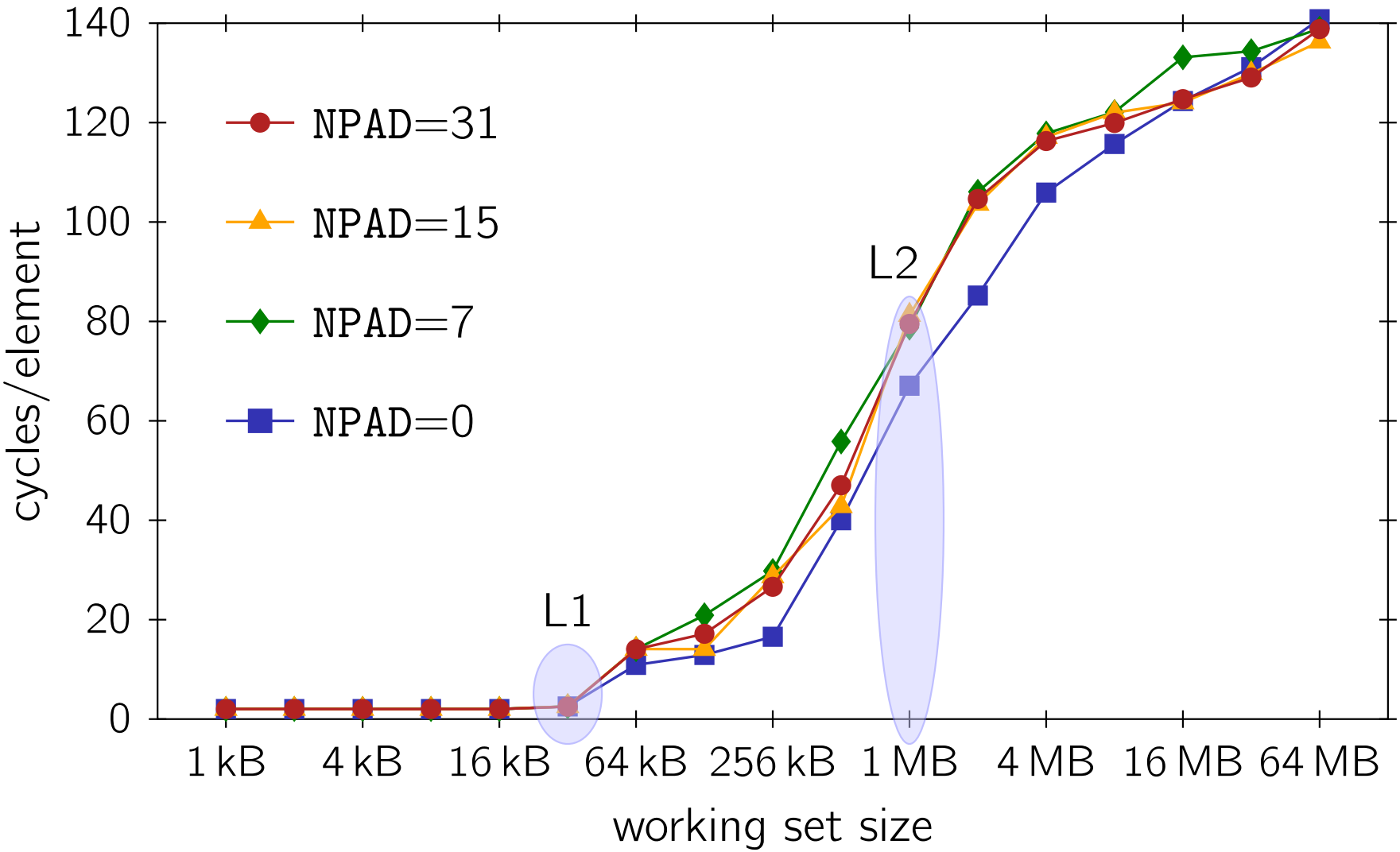
MIPS: Sequential Access — Forward Traversal



MIPS: Sequential Access — Backward Traversal

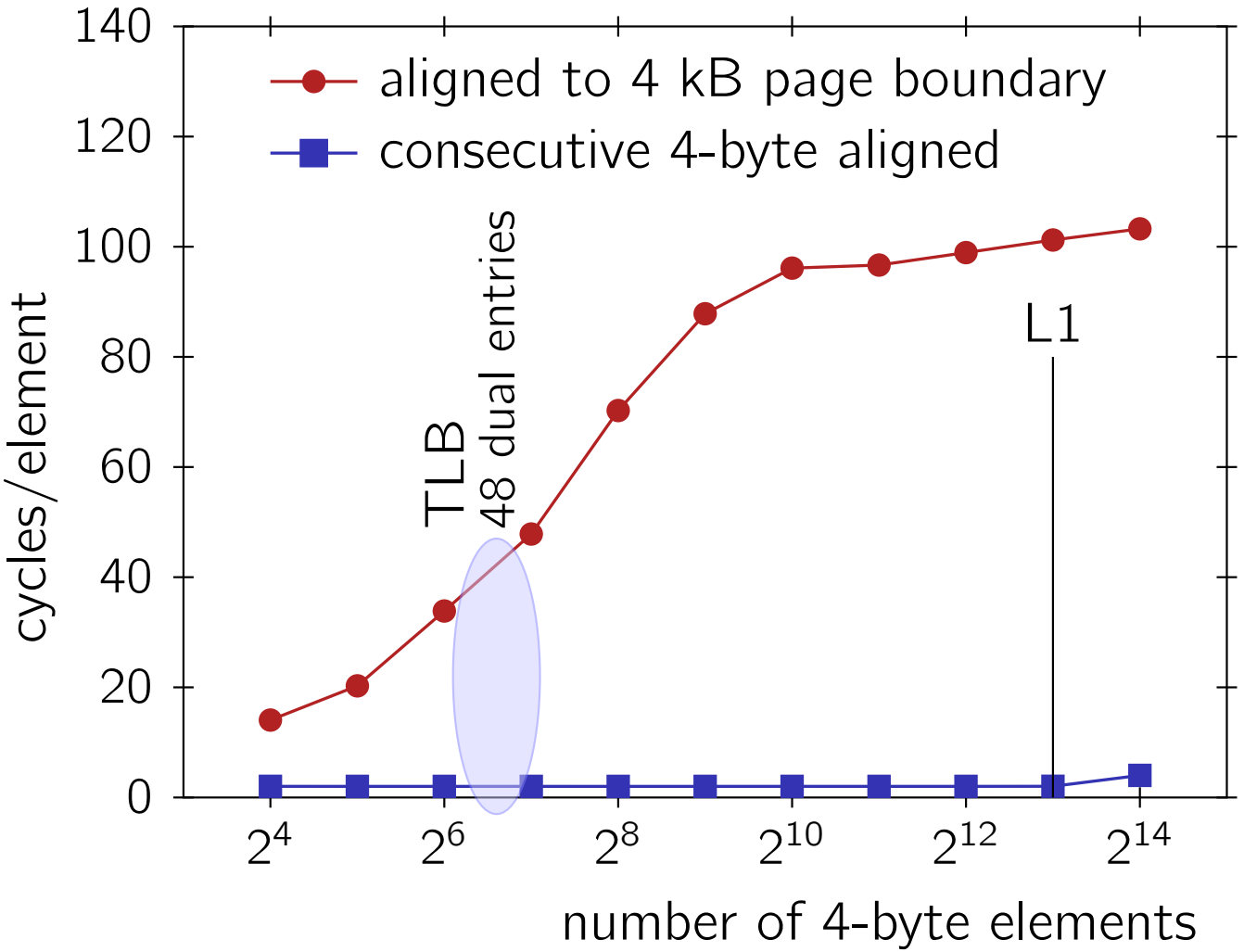


MIPS: Random Access

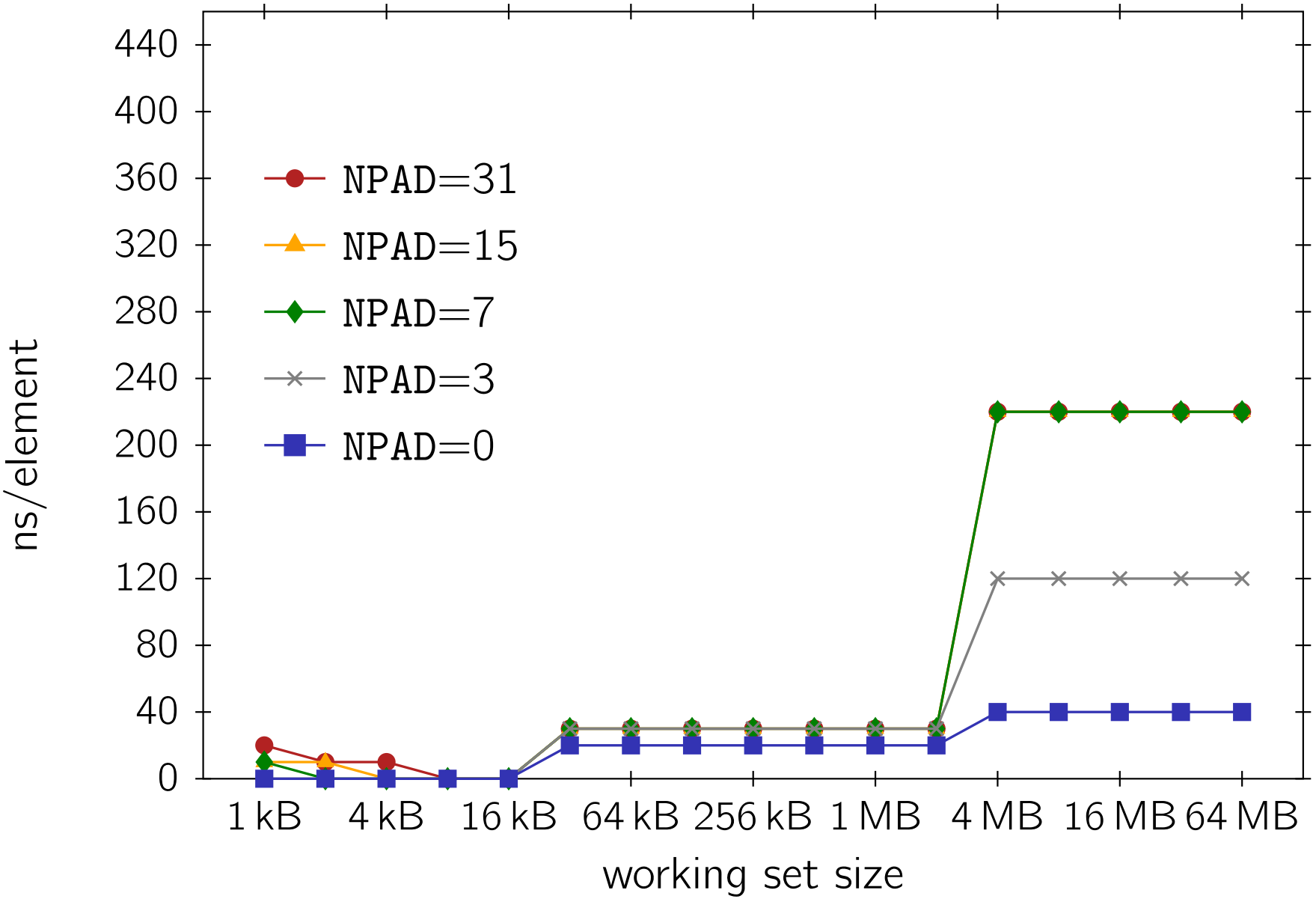


MIPS: TLB Experiment

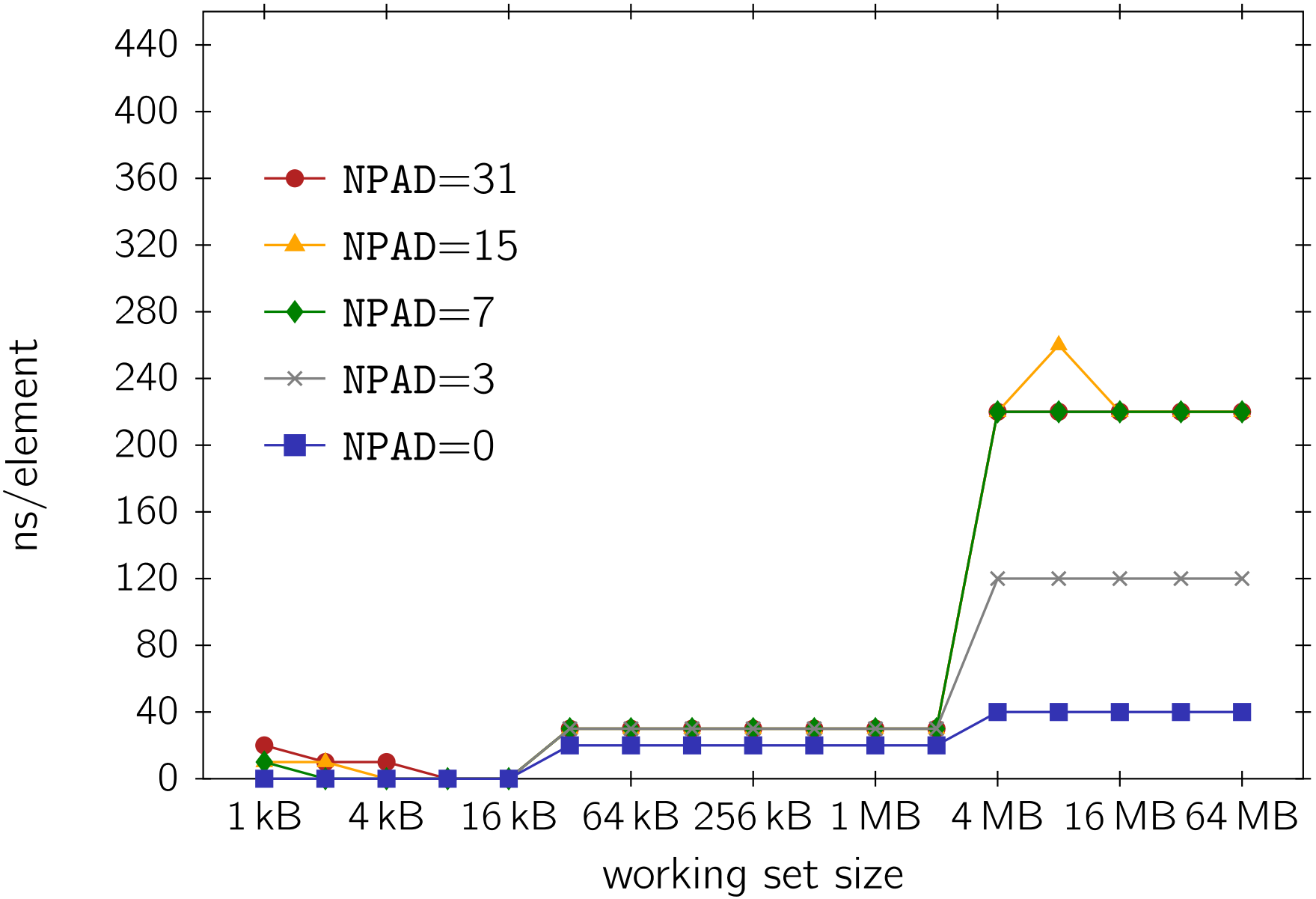
Forward traversal with 4 bytes/element (`NPAD`=0)



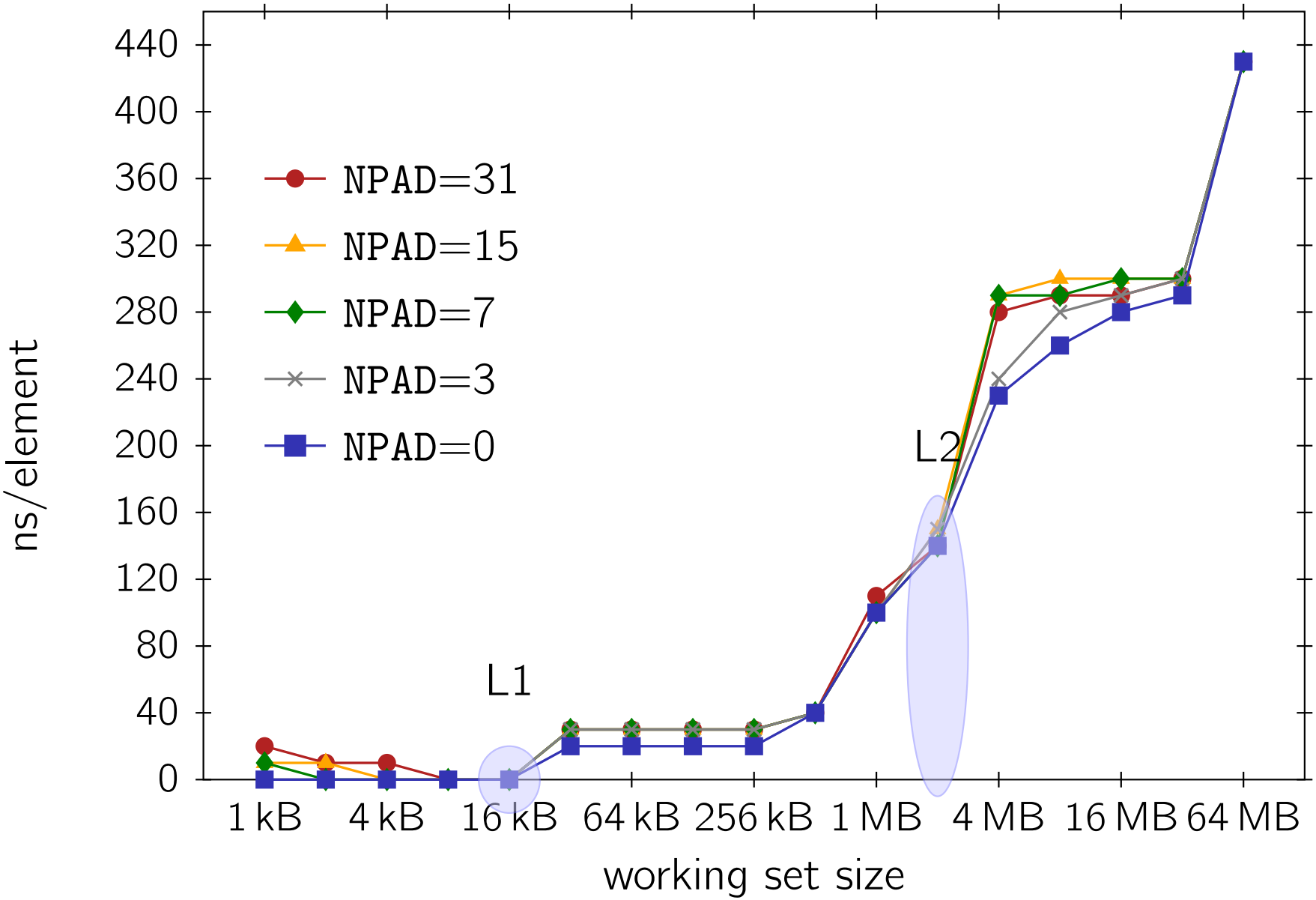
UltraSPARC-IIi : Forward Traversal



UltraSPARC-IIi : Backward Traversal



UltraSPARC-IIi : Random Traversal



UltraSPARC-IIi : Summary

- General
  - 1 core @333 MHz, 64-bit architecture
  - Process : 25 μm
- L1
  - instruction: **16 kB**, cache line (**32 B**)
  - data: **16 kB**, cache line (**32 B**)
- L2
  - external
  - **2 MB**, cache line (**64 B**)