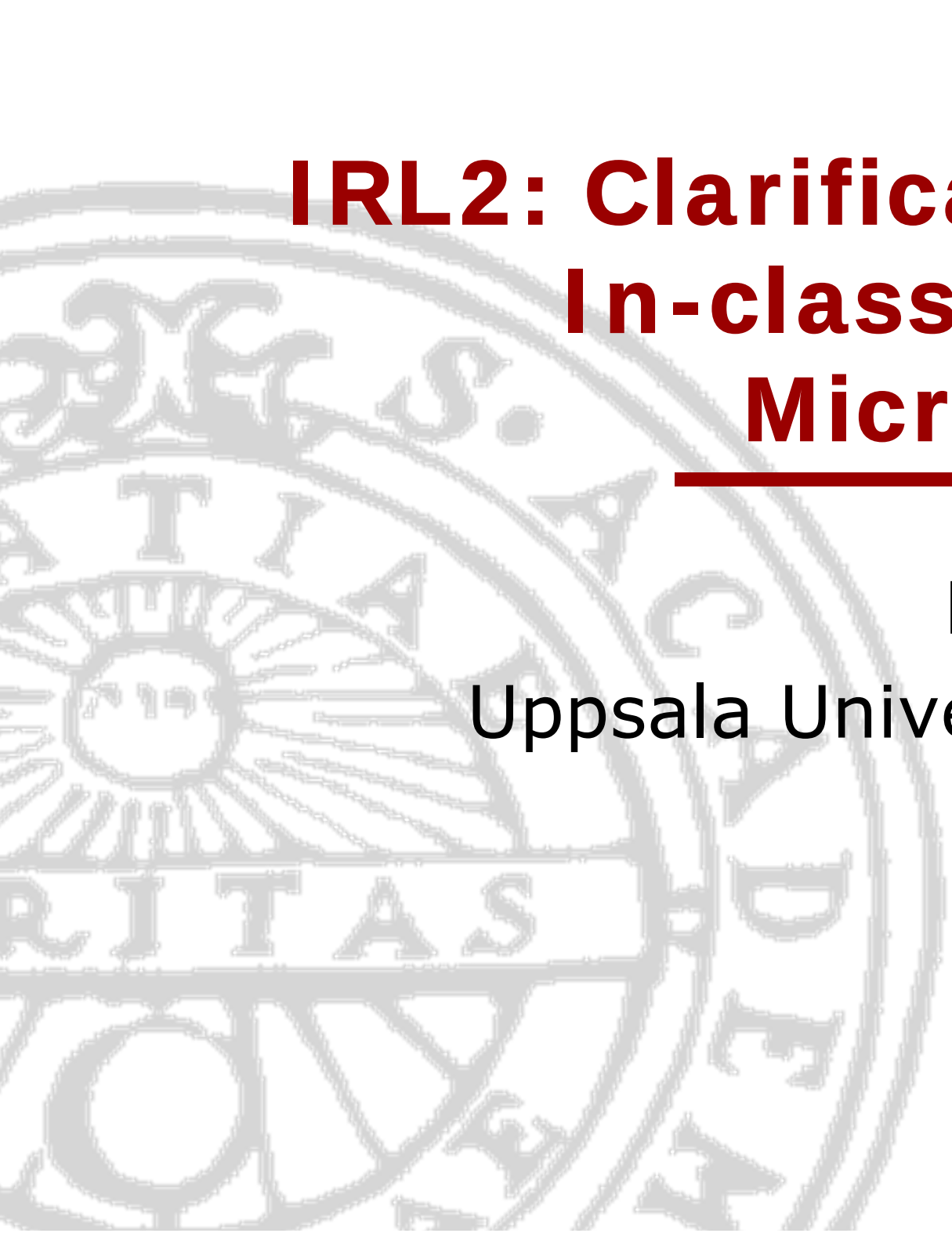


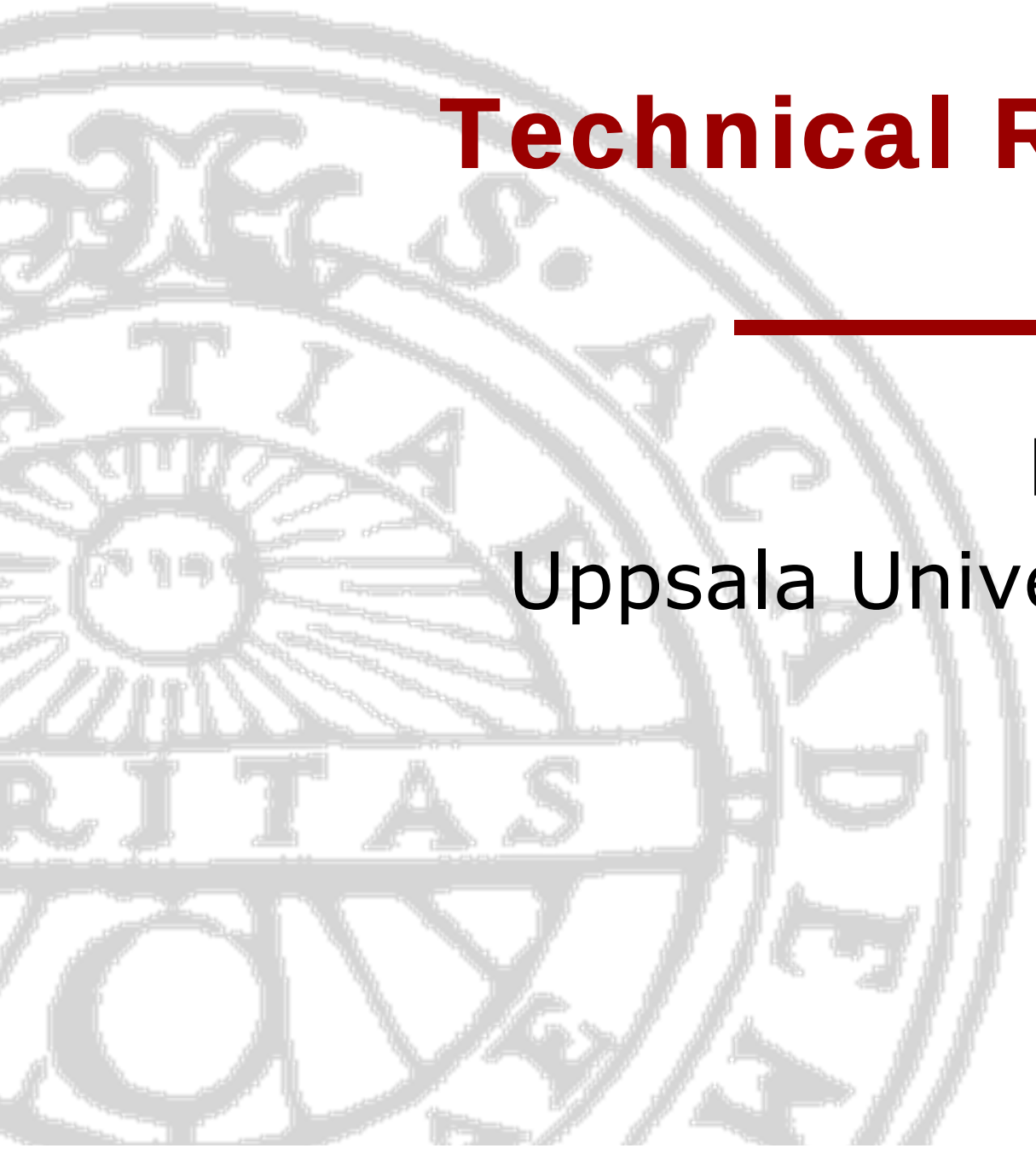
IRL2: Clarification & Issues In-class Cache Design Micro Benchmarks

Erik Hagersten
Uppsala University, Sweden
eh@it.uu.se

UPPSALA UNIVERSITET

Feedback & Quizzes 1st week:

- Still a few technical glitches → David
- Video quizzes: timing and clarity
- Quizz box: size, shape, placement, explanation
- Me: Make video lectures shorter
  - More energy
  - More focus
  - Much clearer questions (focus after next batch "SW-opt")
- Me: I want to be able to reply to video questions (anonymously) and also continuously insert the answer into the video



Technical Remarks on 1st Module

Erik Hagersten
Uppsala University, Sweden
eh@it.uu.se

UPPSALA UNIVERSITET

Why do you miss in a cache

- Mark Hill's three "Cs"
  - Compulsory miss (touching data for the first time)
  - Capacity miss (the cache is too small)
  - Conflict misses (non-ideal cache implementation) (too many names starting with "H")

UPPSALA UNIVERSITET

Why do you miss in a cache

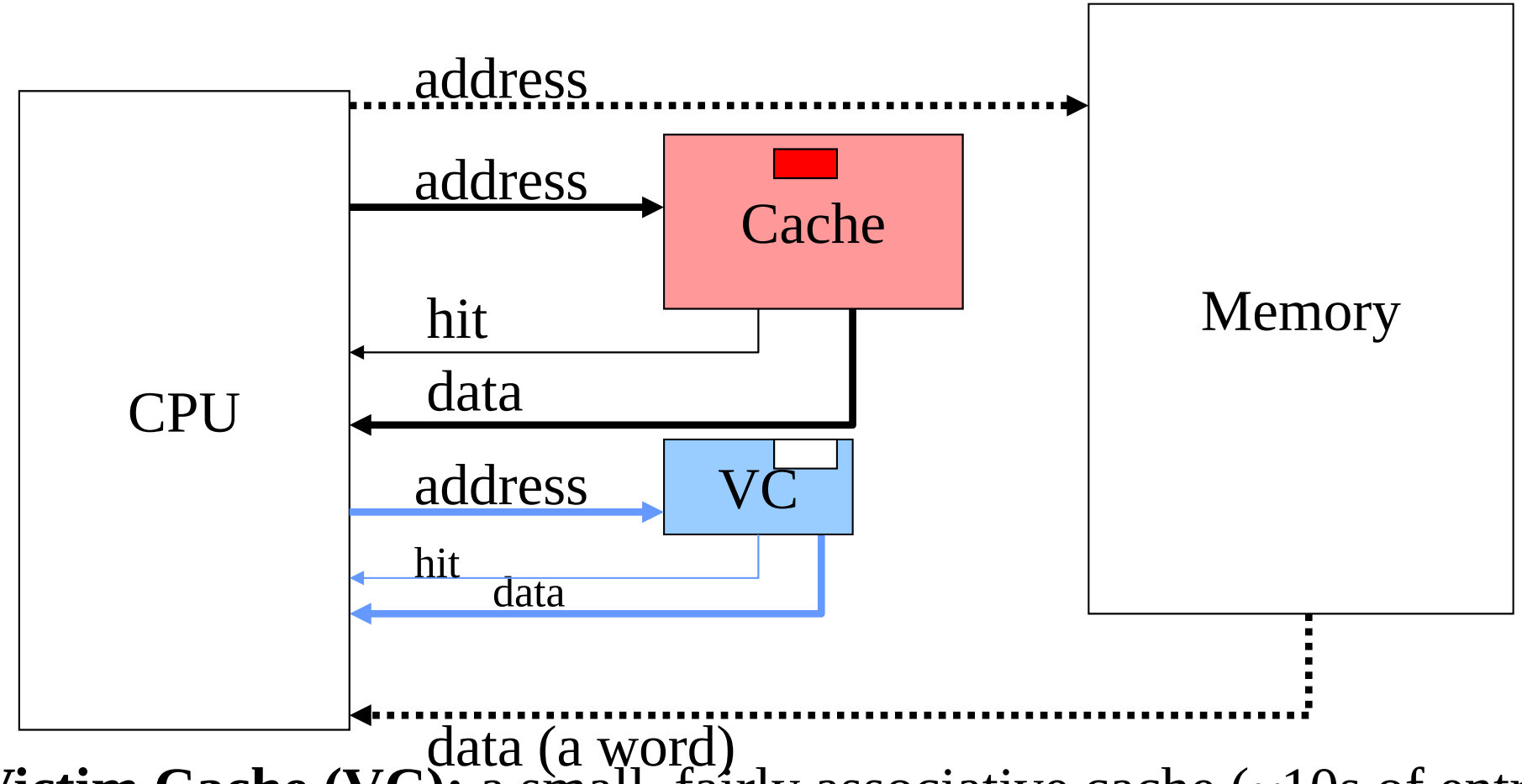
- Rephrasing Mark Hill's three "Cs"
  - Compulsory miss: Miss in an infinitely large (fully associative) cache
  - Capacity miss: Miss in a fully associative with "the same" capacity as this cache, and not a Compulsory miss.
  - Conflict misses: Miss in this cache and none of the other miss types.

AVDARK 2013

UPPSALA UNIVERSITET

What miss type is primarily removed by:

- Large cache line? (while the capacity and associativity stays the same)?
- Associativity? (everything else stays the same…)
- Victim cache? (adding VC with 16 cache lines, fully associative)
- 2-way skewed cache? (compared with a 2-way associative)



Victim Cache (VC): a small, fairly associative cache (~10s of entries)

Cache lookup: search cache and VC in parallel

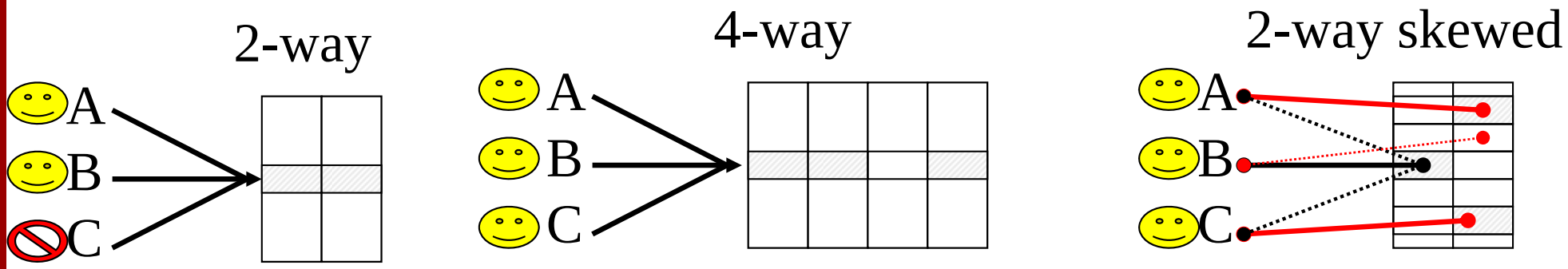
Cache replacement: move victim to the VC and replace from VC

VC hit: swap VC data with the corresponding data in Cache

"A second life ☺"

Skewed Associative Cache

Example: A, B and C have a three-way conflict

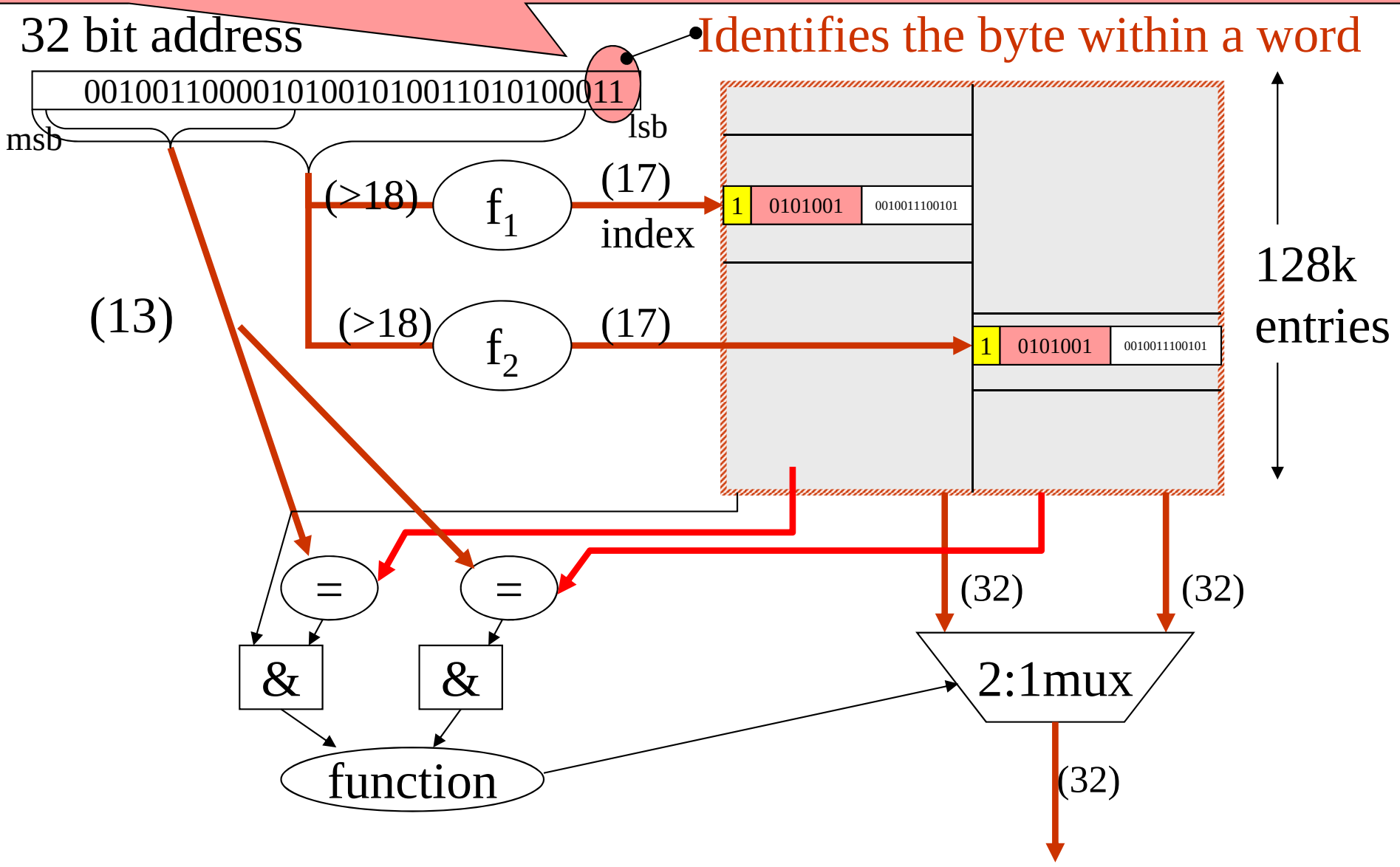


- It has been shown that 2-way skewed performs roughly the same as 4-way caches

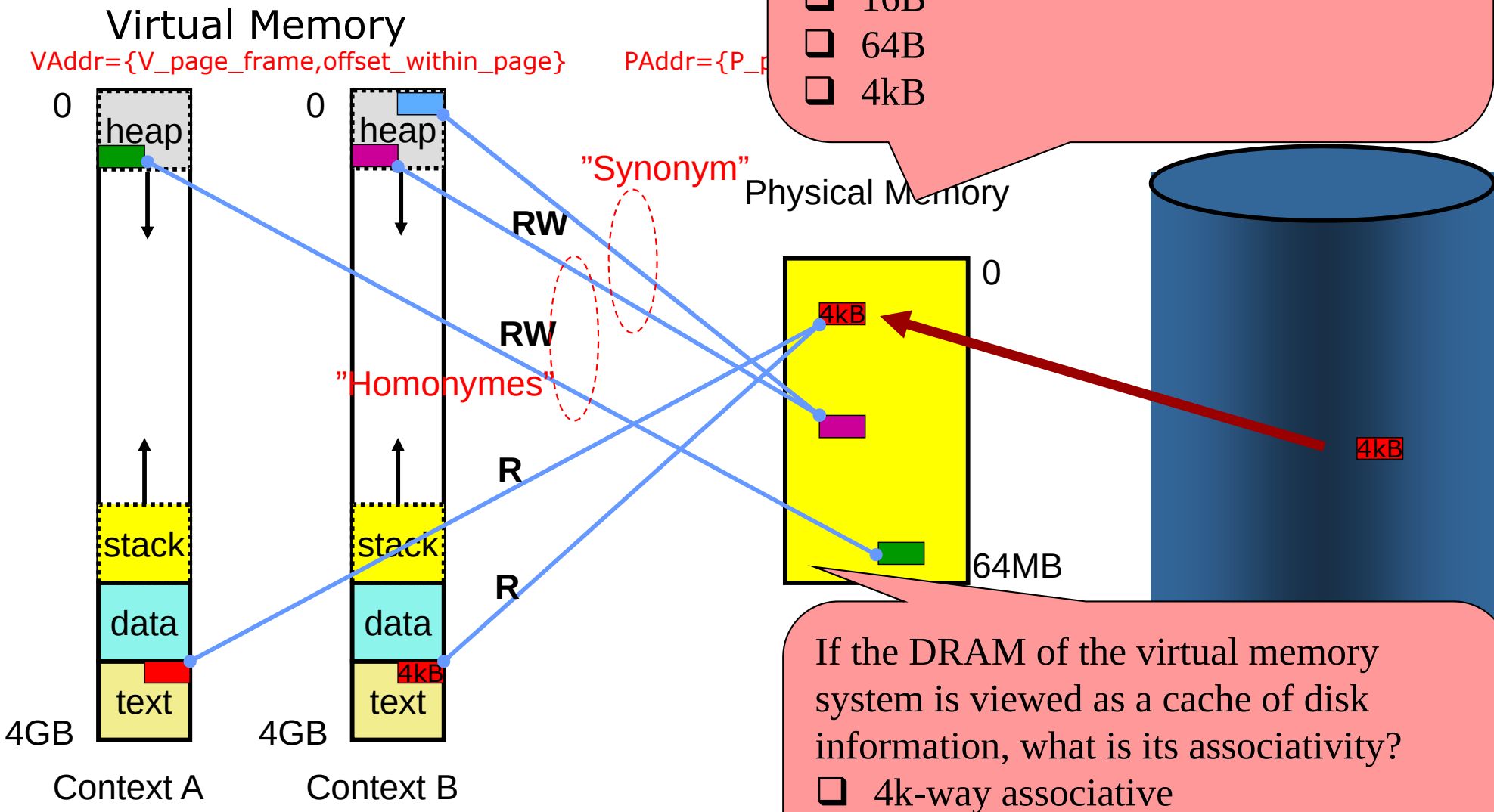
- Uses less power than 4-way

Why does a 2-way skewed cache use less power than a 4-way cache [of the same size]?

- It has much fewer SRAM bits in total (Static energy)
- It reads fewer bits from the SRAM on a cache lookup (Dynamic energy)
- It works at a lower frequency



Translation & I



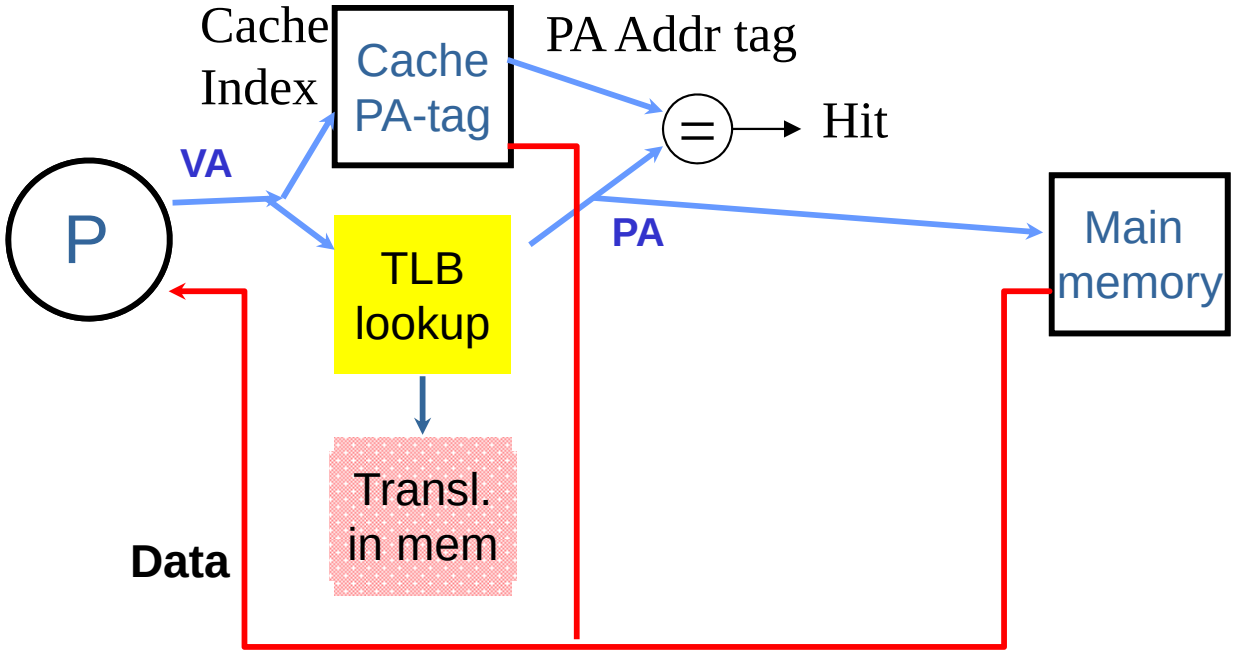
If the DRAM of the virtual memory system is viewed as a cache of disk information, how large are its cache lines??

- ❑ 16B
- ❑ 64B
- ❑ 4kB

If the DRAM of the virtual memory system is viewed as a cache of disk information, what is its associativity?

- ❑ 4k-way associative
- ❑ Direct mapped
- ❑ Fully associative

Virtually Indexed Physically Tagged =VIPT



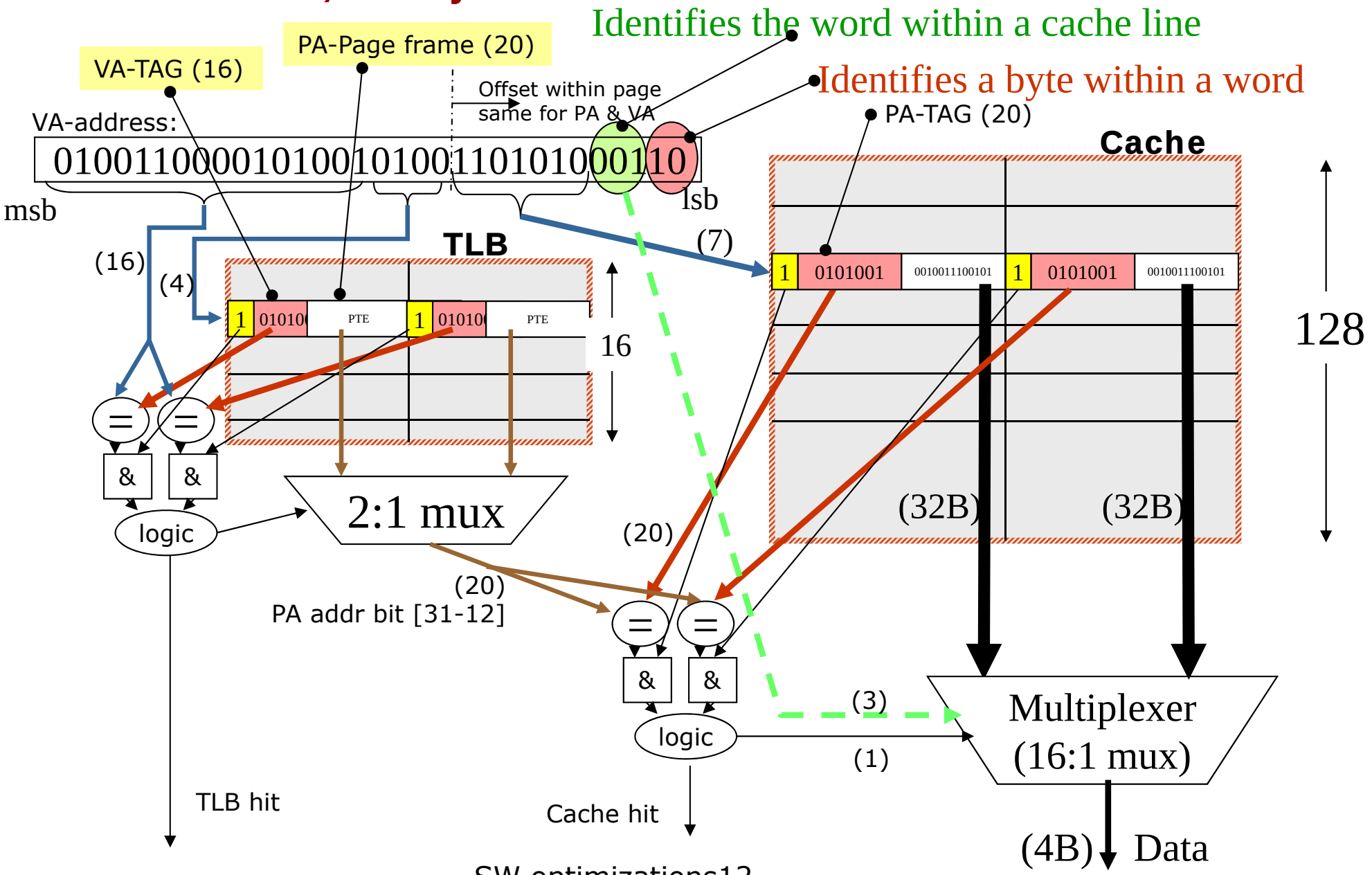
Have to guarantee that all aliases have the same index

- VIPT_cache_size ≤ (page-size * associativity)
- (Page coloring can help further)

Putting it all together: VIPT

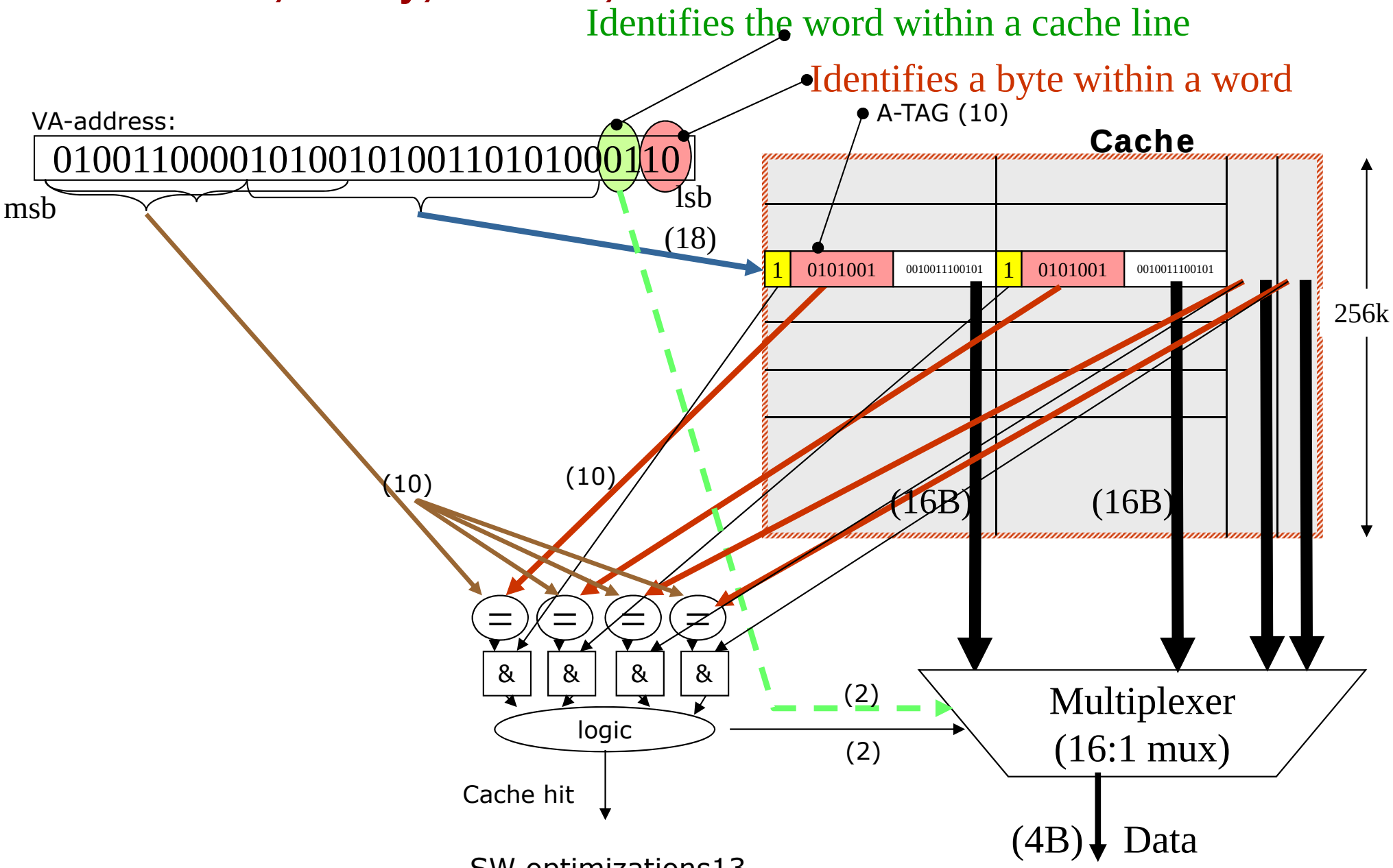
Cache: 8kB, 2-way, CL=32B, word=4B, page =4kB

TLB: 32 entries, 2-way



In-Class: Design a cache

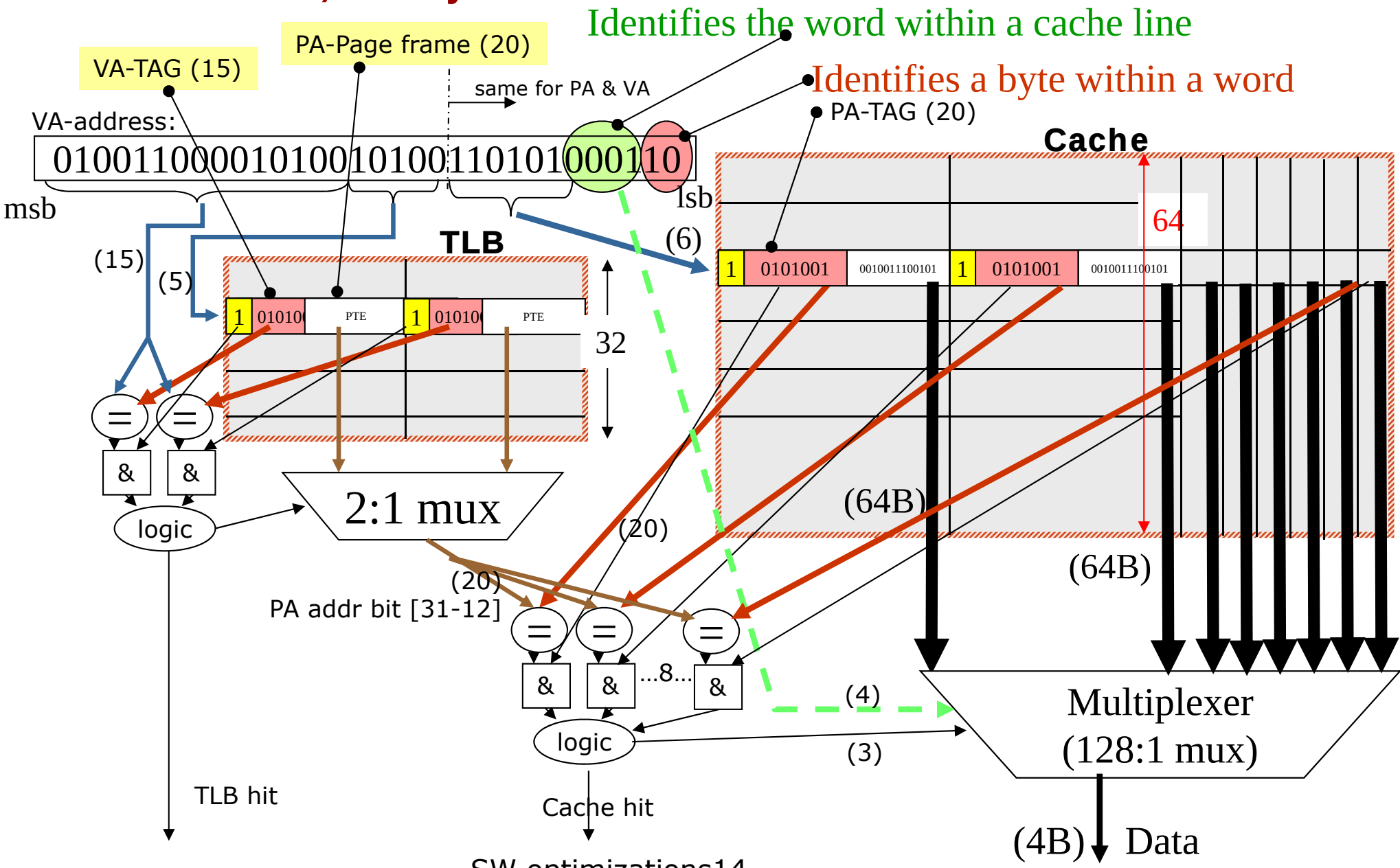
Cache: 16MB, 4-way, CL=16B, word=4B



In-class: Design a VIPT

Cache: 32kB, maximum-way, CL=64B, word=4B, page =4kB

TLB: 64 entries, 2-way



UPPSALA UNIVERSITET

Micro Benchmark Signature

```
for (times = 0; times < Max; times++) /* many times*/

  for (i=0; i < ArraySize; i = i + Stride)
    dummy = A[i]; /* touch an item in the array */
```

Measuring the averge access time to memory, while varying ArraySize and Stride, will allow us to reverse-engineer the memory system.
(need to turn off HW prefetching...)

AVDARK 2013

Stepping through the array

```
for (times = 0; times < Max; times++) /* many times*/

  for (i=0; i < ArraySize; i = i + Stride)
    dummy = A[i]; /* touch an item in the array */
```

0

Array Size = 16, Stride=4

0

Array Size = 16, Stride=8…

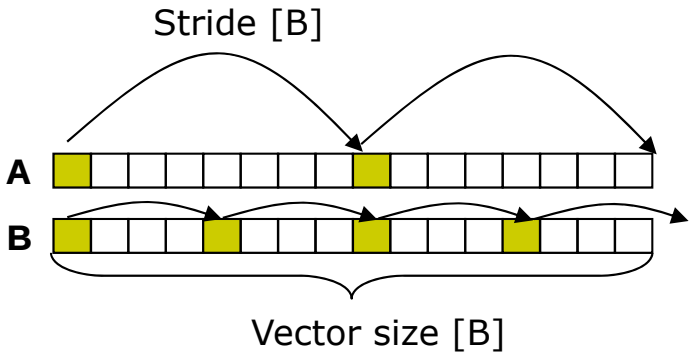
0

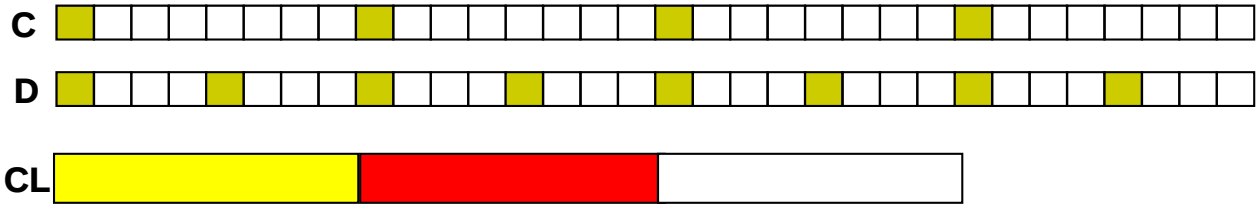
Array Size = 32, Stride=4…

0

Array Size = 32, Stride=8…

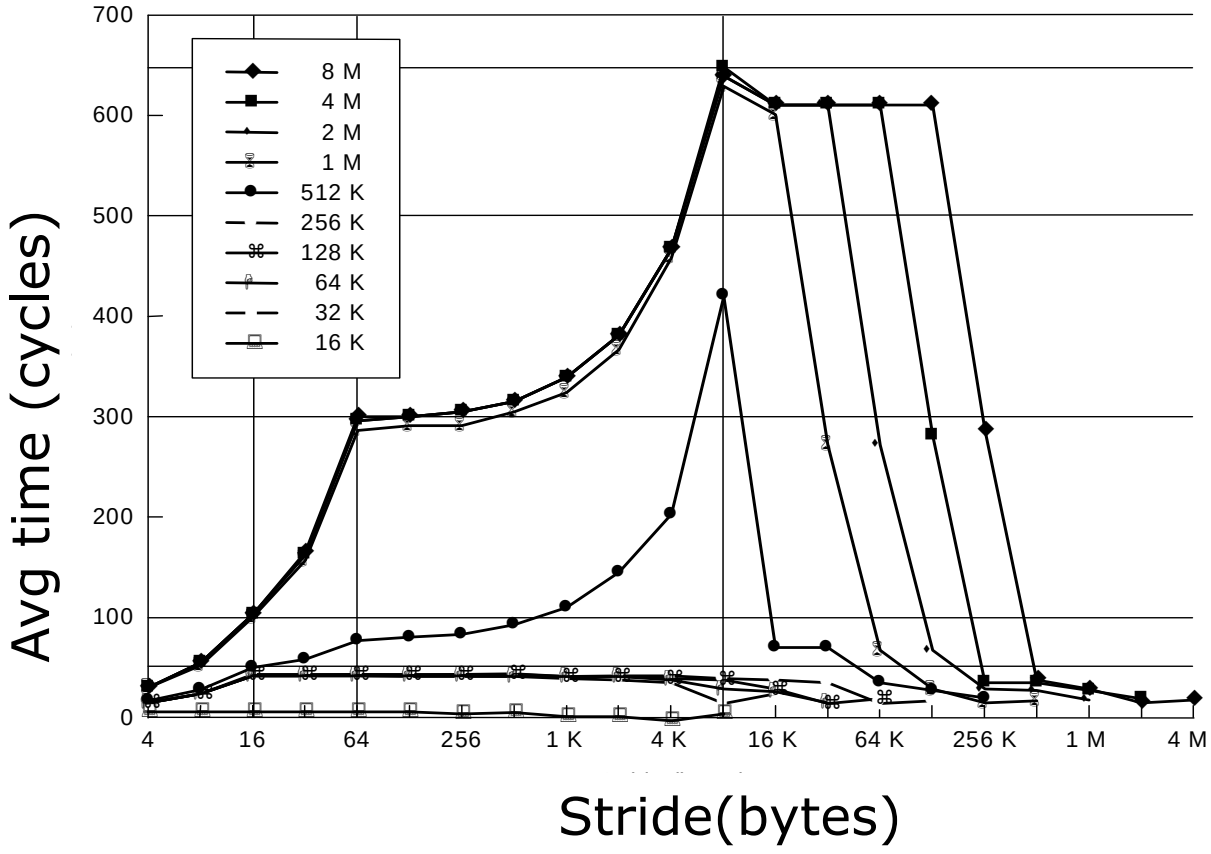
Stepping through the array





Micro Benchmark Signature

```
for (times = 0; times < Max; times++) /* many times*/

  for (i=0; i < ArraySize; i = i + Stride)
    dummy = A[i]; /* touch an item in the array */
```



L1: size=4kB CL= 32B

L2: size=256kB CL= 128B 128way

Page: size=16kB Memory: Lat = 220

TLB: reach=1MB #Entries= 64

Microbenchmark

Why is it falling for strides >16kB?

Why is it hurting 2MB and not 1MB?

This ski-slope of the "next next" curve" gives the CL size for ???

This ski-slope of the "next" curve" gives the CL size for L2

This ski-slope of the "L2 curve" gives the CL size for L1

Why is it falling?

Latency

Vector size

2MB

1MB

512kB

256kB
128kB
64kB
32kB
16kB
8kB

Vector=4kB

Stride [B]

Strides

In-class: Guess the cache

L1: size= 32kB CL= 32B

L2: size= 1MB CL= 64B

Page: size=4kB

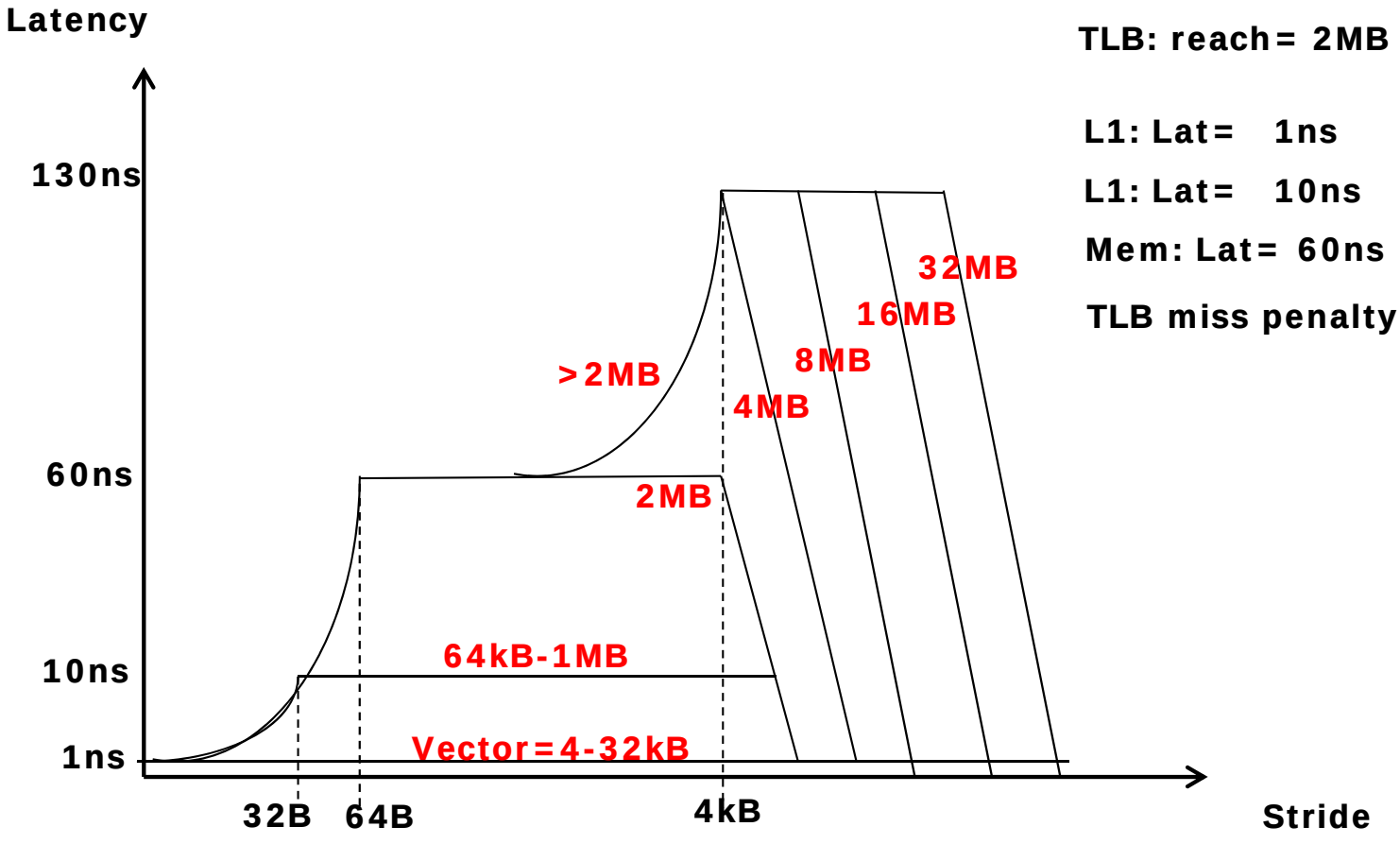
TLB: reach= 2MB #Entries= 512

L1: Lat= 1ns

L1: Lat= 10ns

Mem: Lat= 60ns

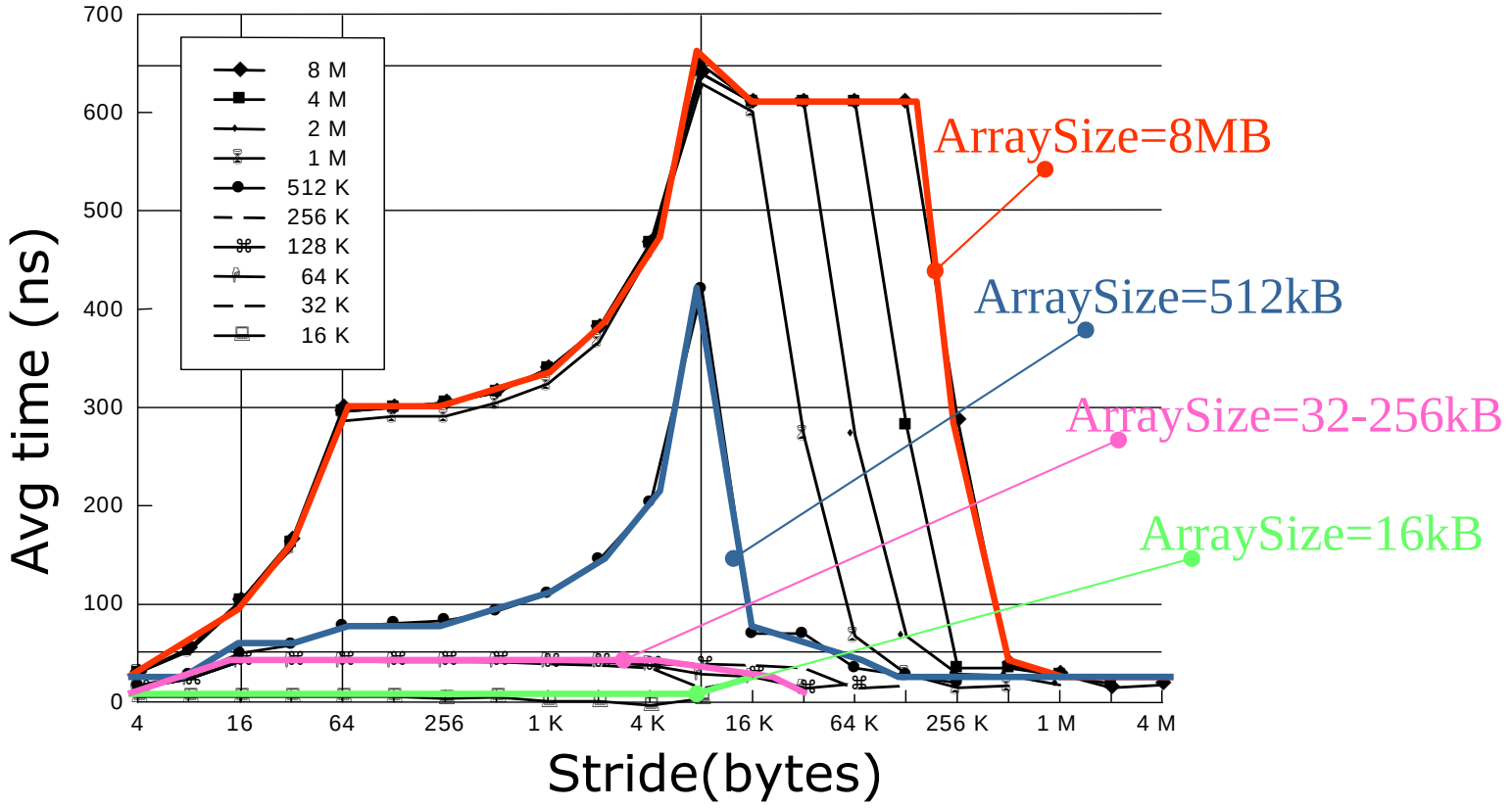
TLB miss penalty: 70ns



AVDARK 2013

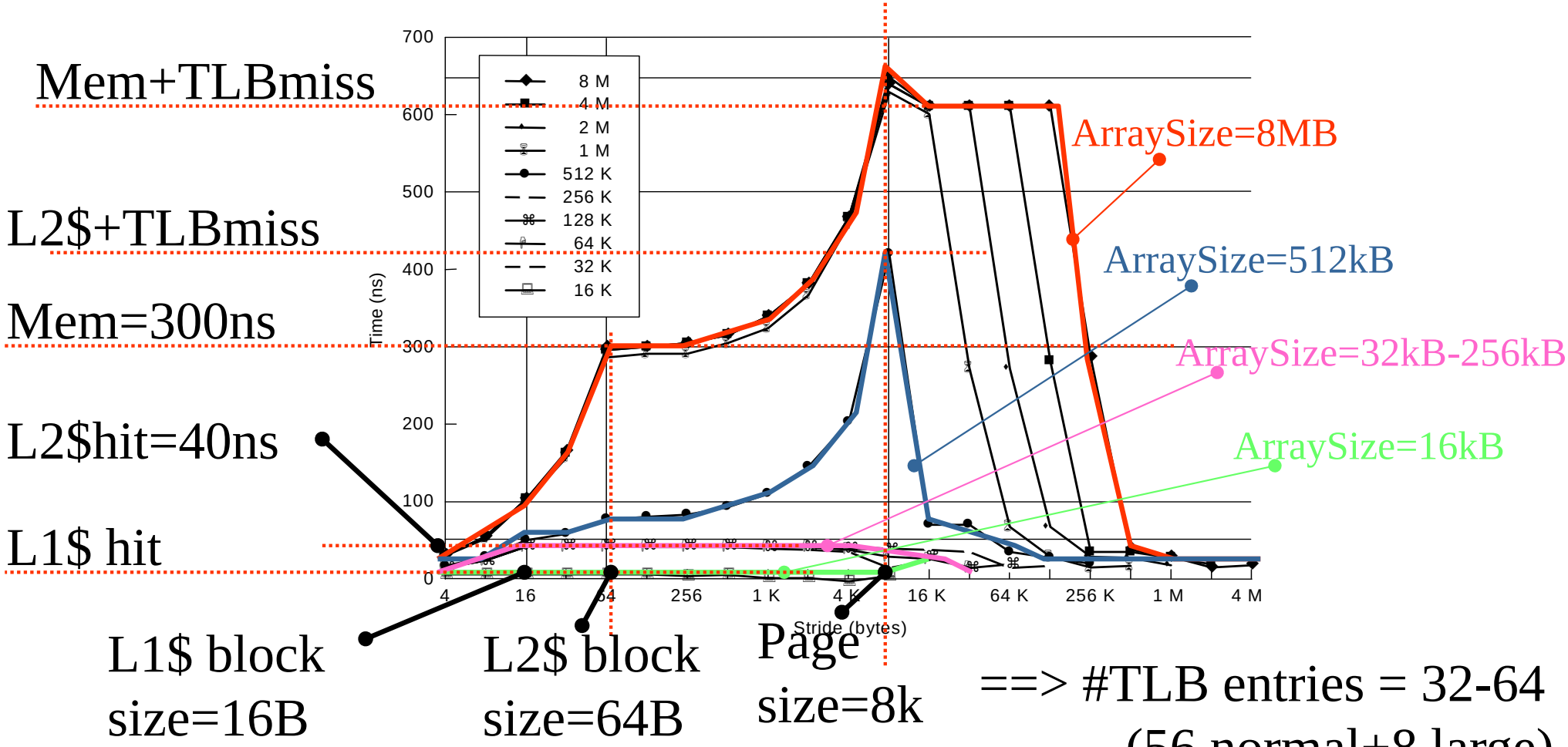
Micro Benchmark Signature

```
for (times = 0; times < Max; time++) /* many times*/

  for (i=0; i < ArraySize; i = i + Stride)
    dummy = A[i]; /* touch an item in the array */
```



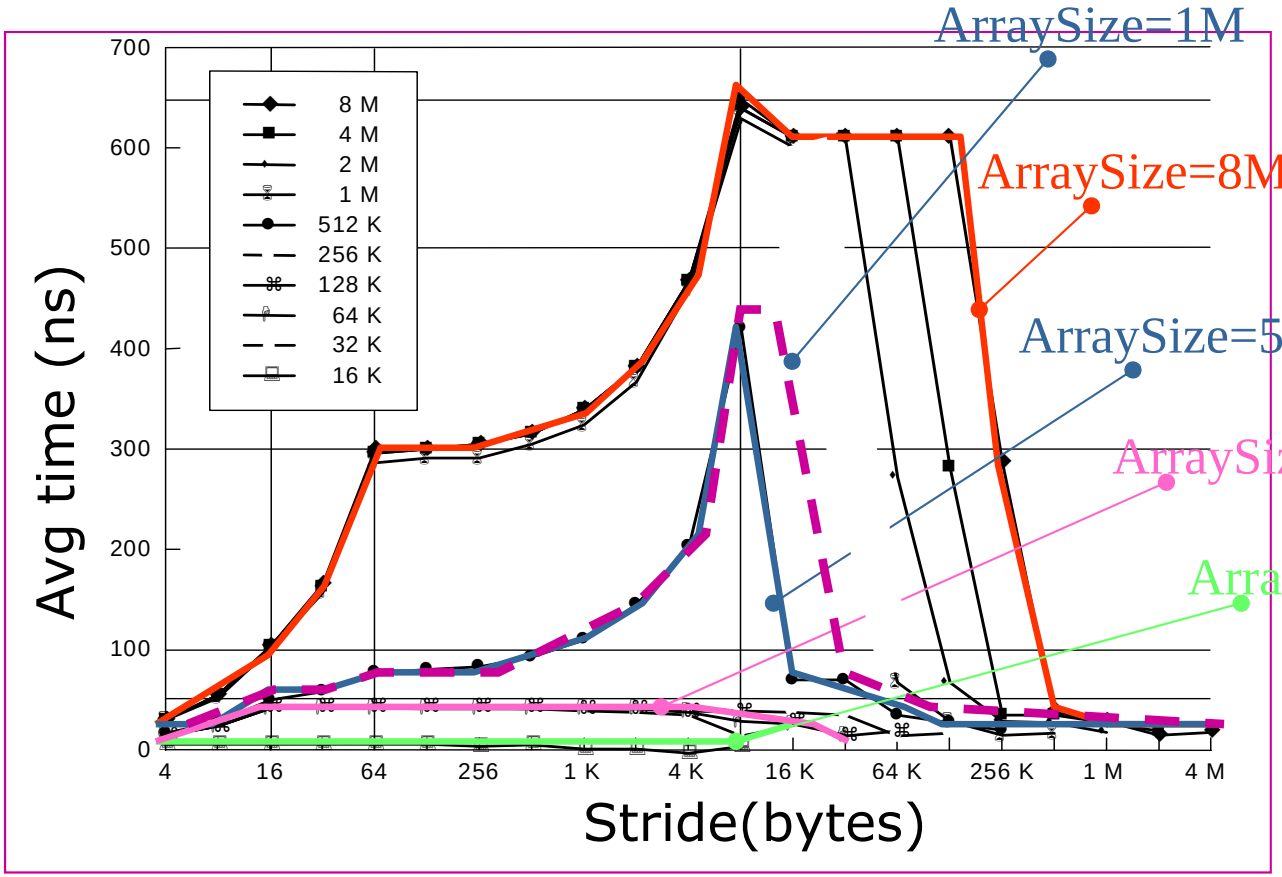
Micro Benchmark Signature

```
for (times = 0; times < Max; time++) /* many times*/

  for (i=0; i < ArraySize; i = i + Stride)
    dummy = A[i]; /* touch an item in the array */
```



Twice as large L2 cache ???

```
for (times = 0; times < Max; time++) /* many times*/

  for (i=0; i < ArraySize; i = i + Stride)
    dummy = A[i]; /* touch an item in the array */
```



Twice as large TLB…

```
for (times = 0; times < Max; time++) /* many times*/

  for (i=0; i < ArraySize; i = i + Stride)
    dummy = A[i]; /* touch an item in the array */
```