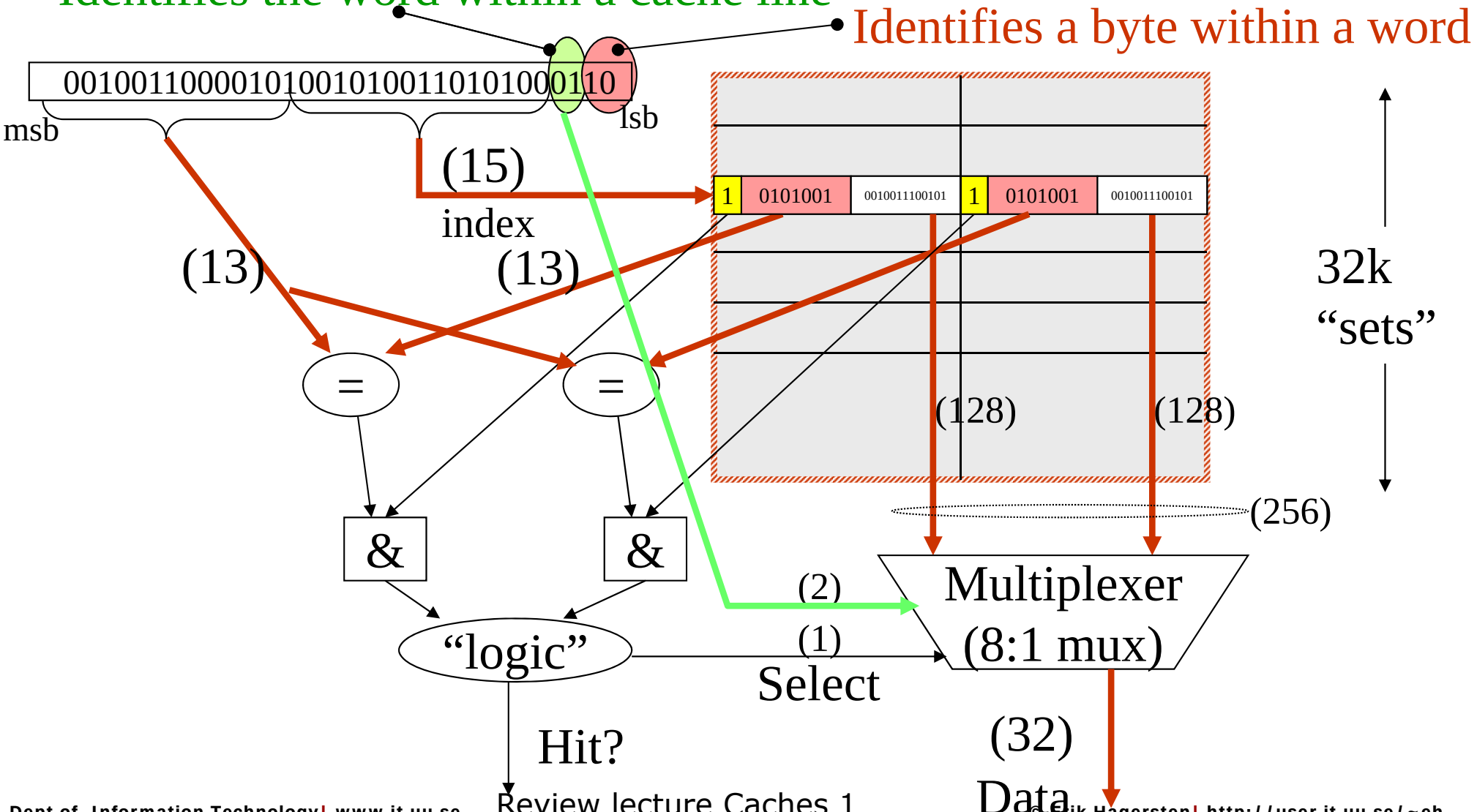
In-Class: Design a cache

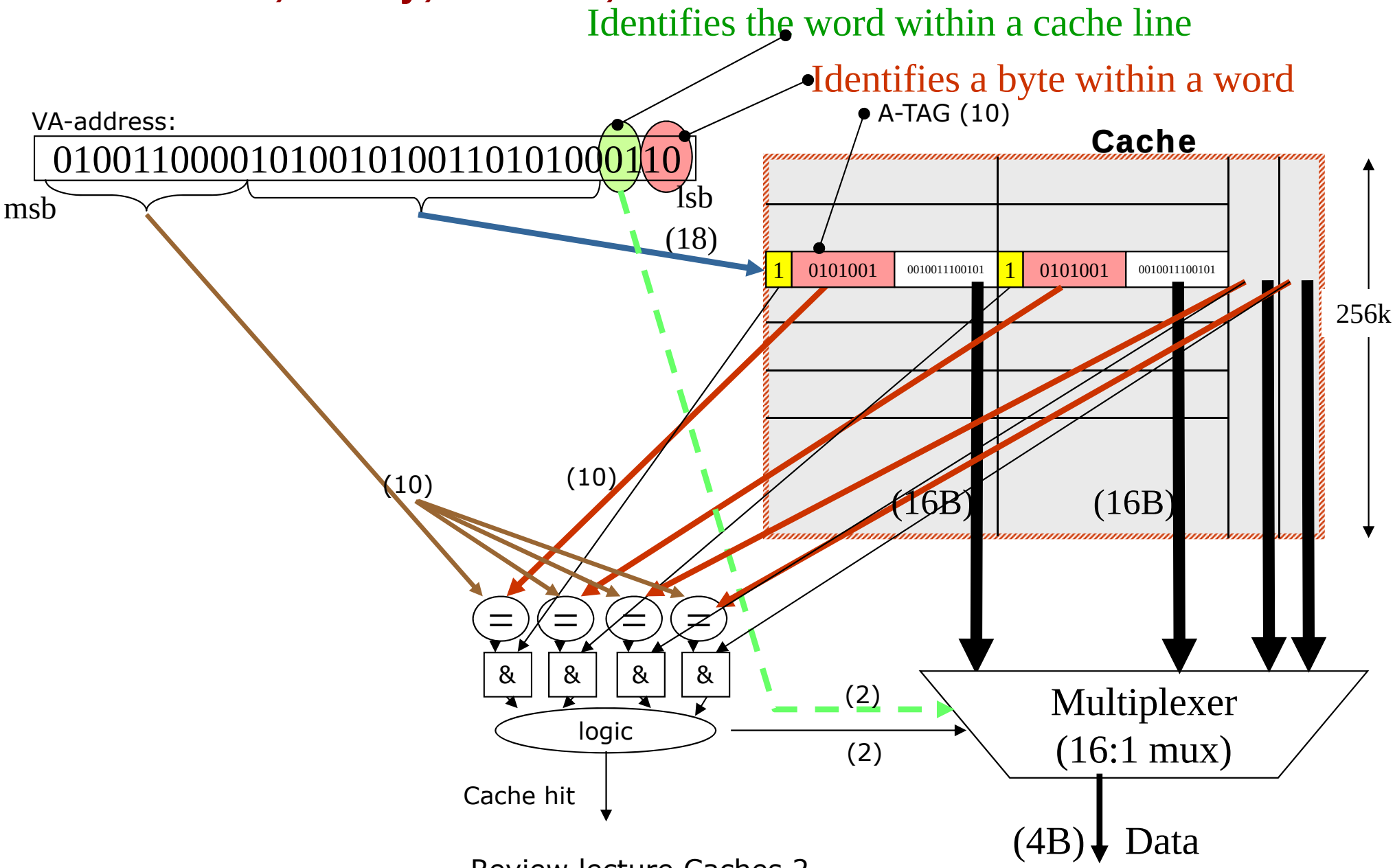
Cache: 16MB, 4-way, CL=16B, word=4B

This is a completely different cache design!

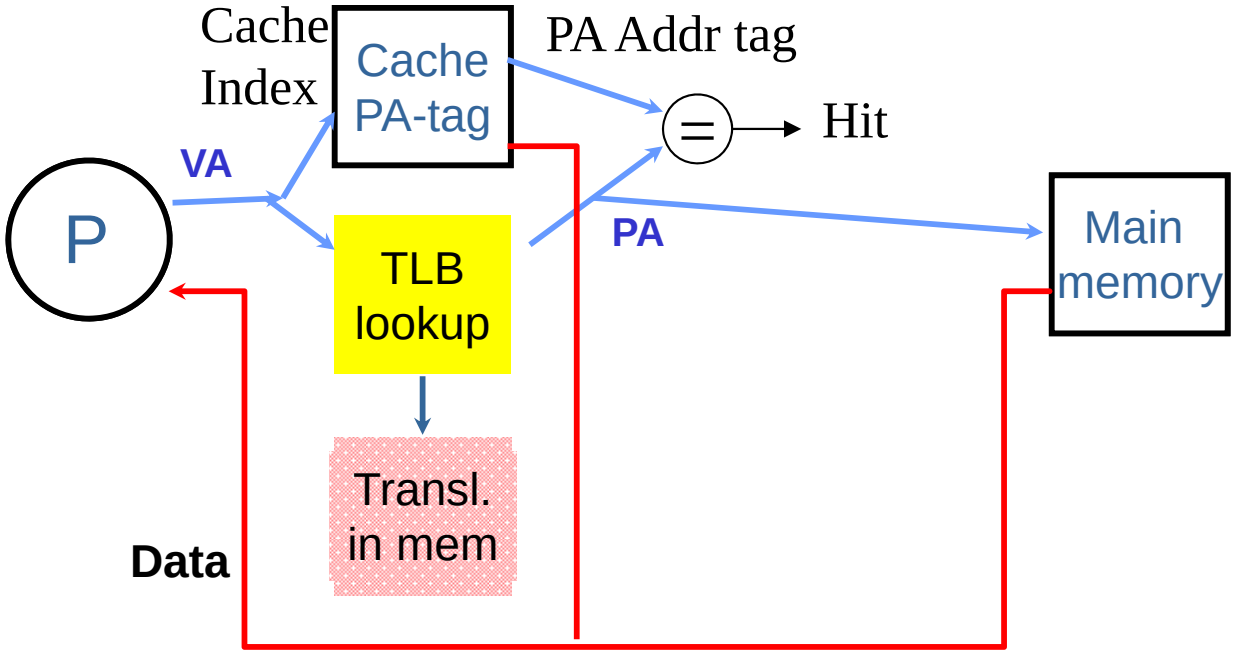


In-Class: Design a cache

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



Virtually Indexed Physically Tagged =VIPT



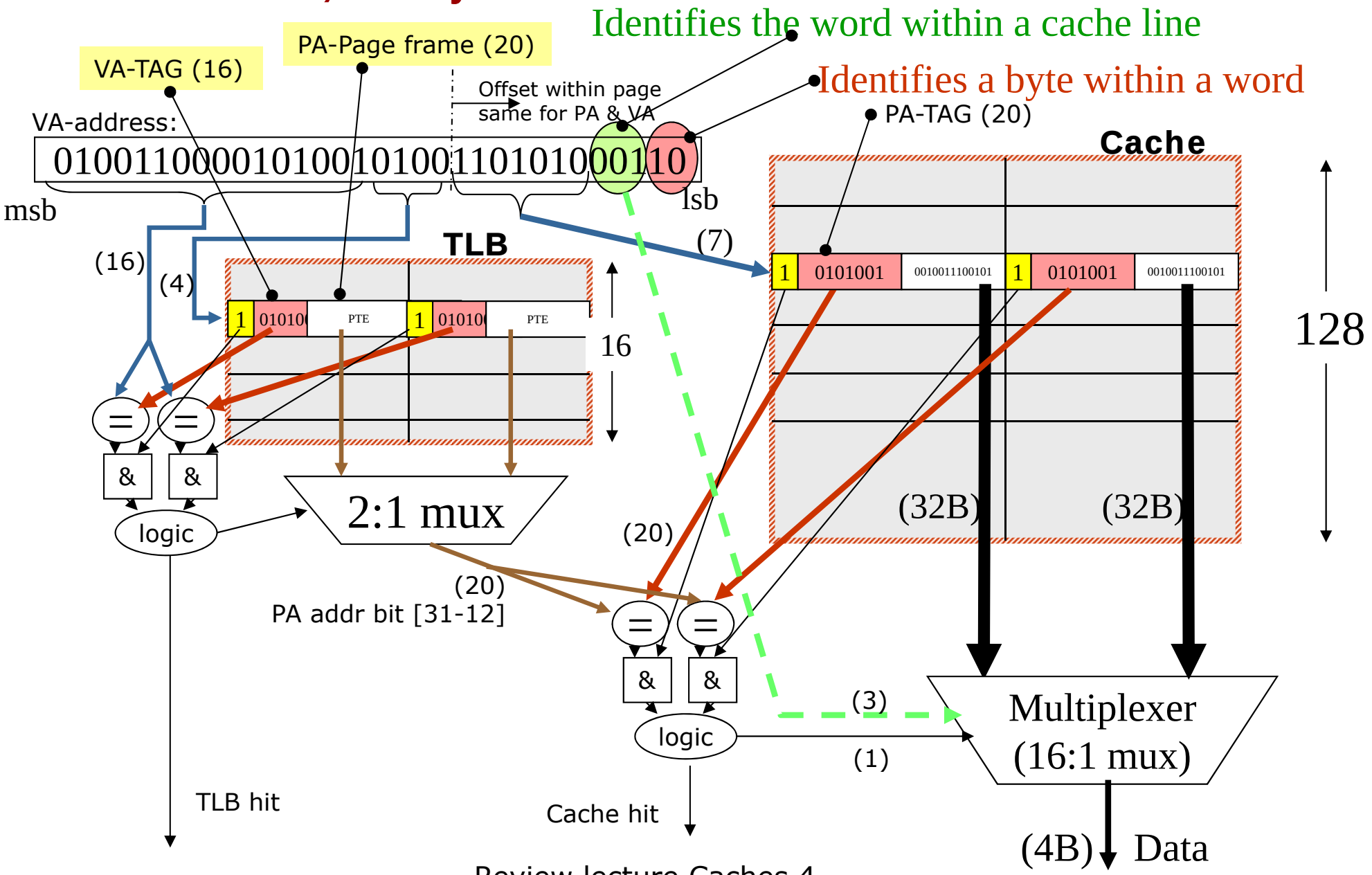
Have to guarantee that all aliases have the same index

- 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

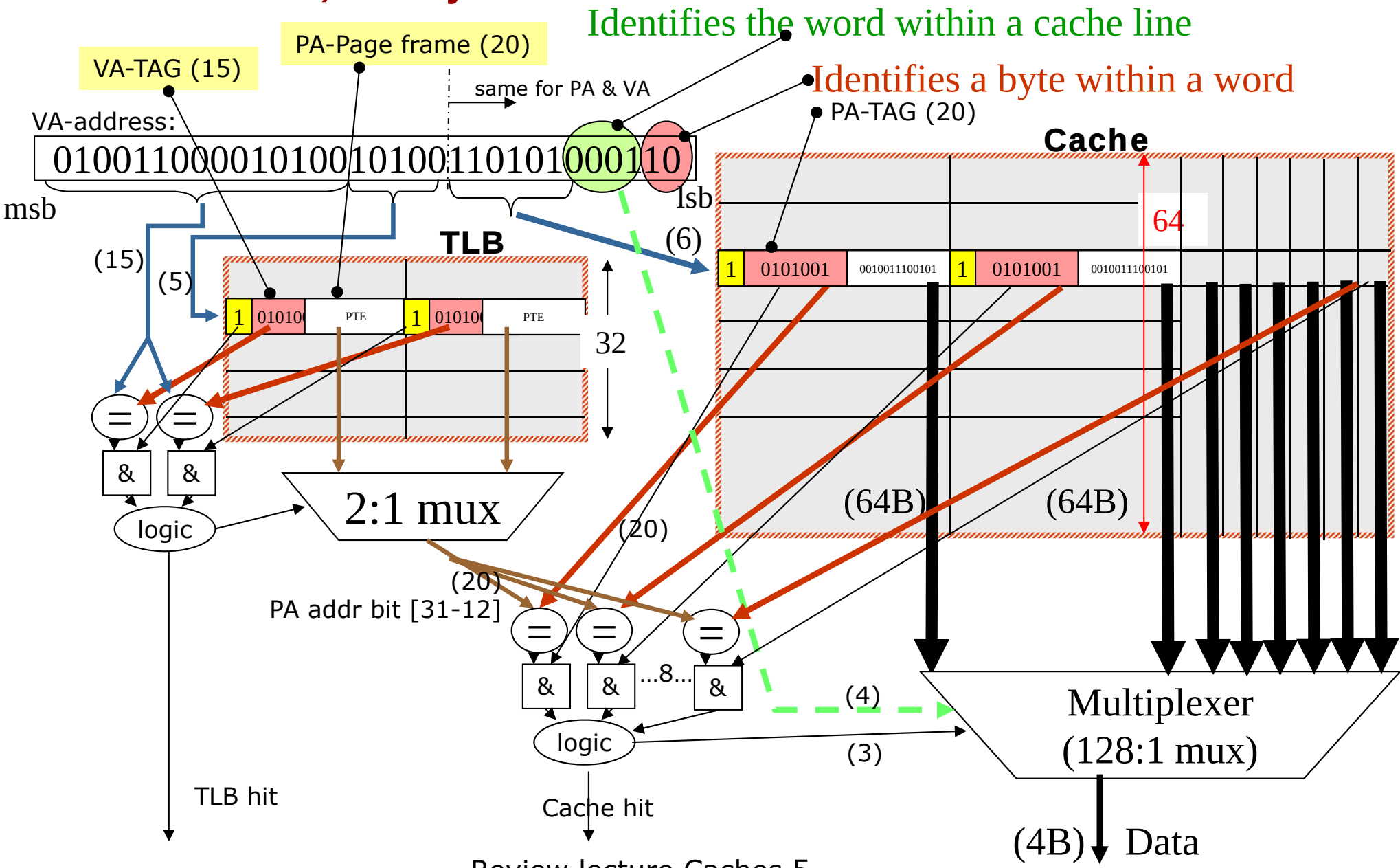
TLB: 32 entries, 2-way



In-class: Design a 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...)

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 */
```

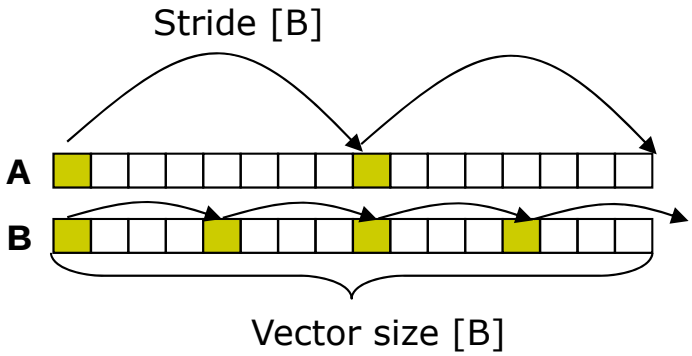
0 Array Size = 16, Stride=4

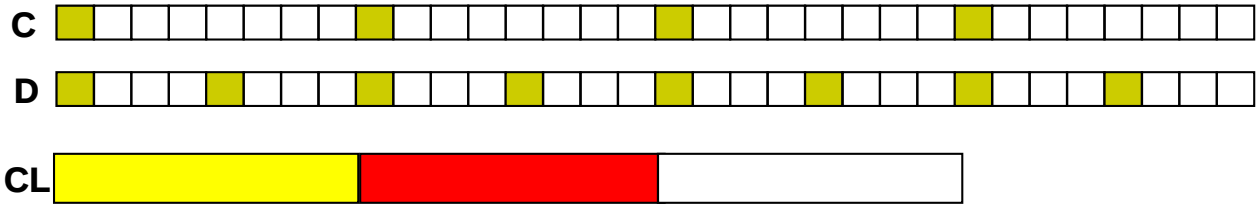
0 Array Size = 16, Stride=8…

0 Array Size = 32, Stride=4…

0 Array Size = 32, Stride=8…

Stepping through the array

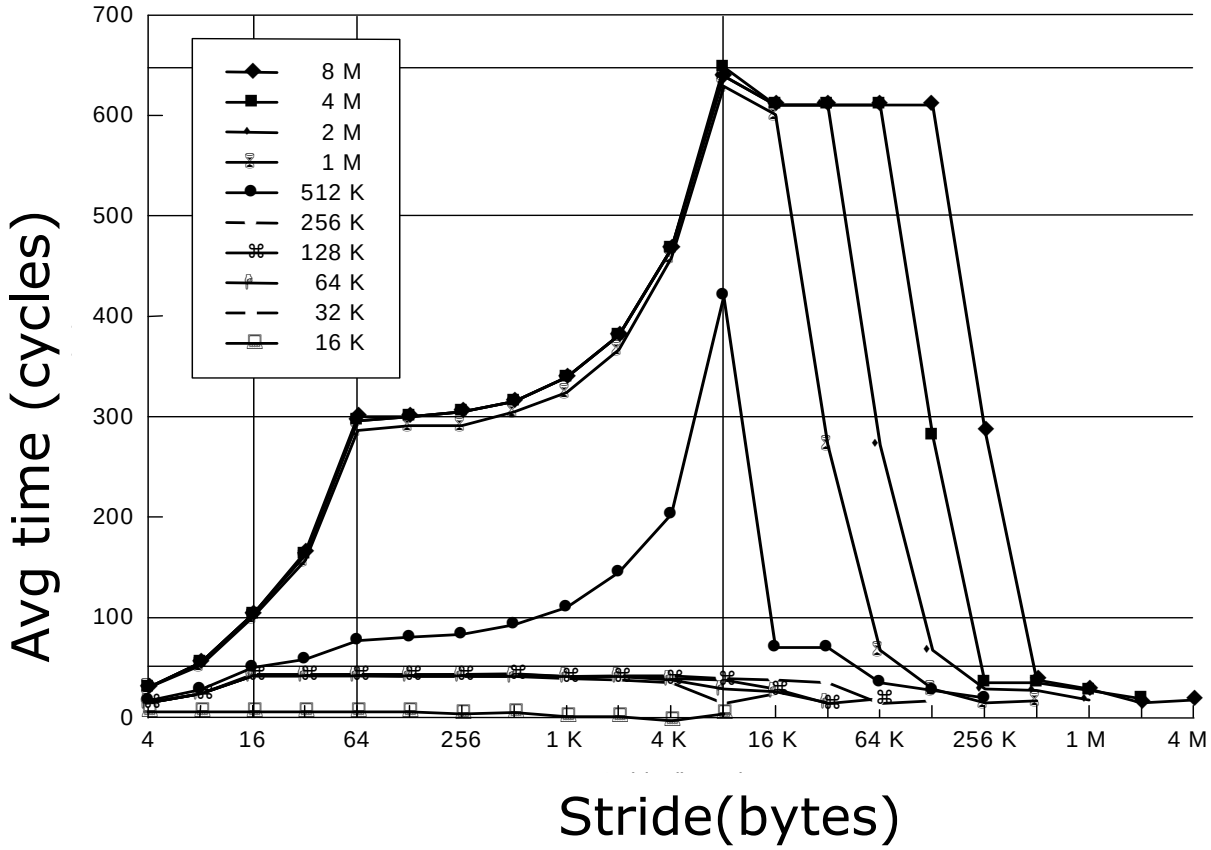




Micro Benchmark Signature

```
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?

2MB

1MB

512kB

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

Vector=4kB

Latency

Vector size

4k
8k
16k
32k
64k
128k
256k
512k
1M
2M
4M
8M

700
600
500
300
100
0

4 8 16 32 64 128 256 512 1k 2k 4k 8k 16k 32k 64k 128k 256k

Stride [B]

Strides

In-class: Guess the Cache

L1: size= 32kB CL= 32B

L2: size= 1MB CL= 64B

Page: size=4kB

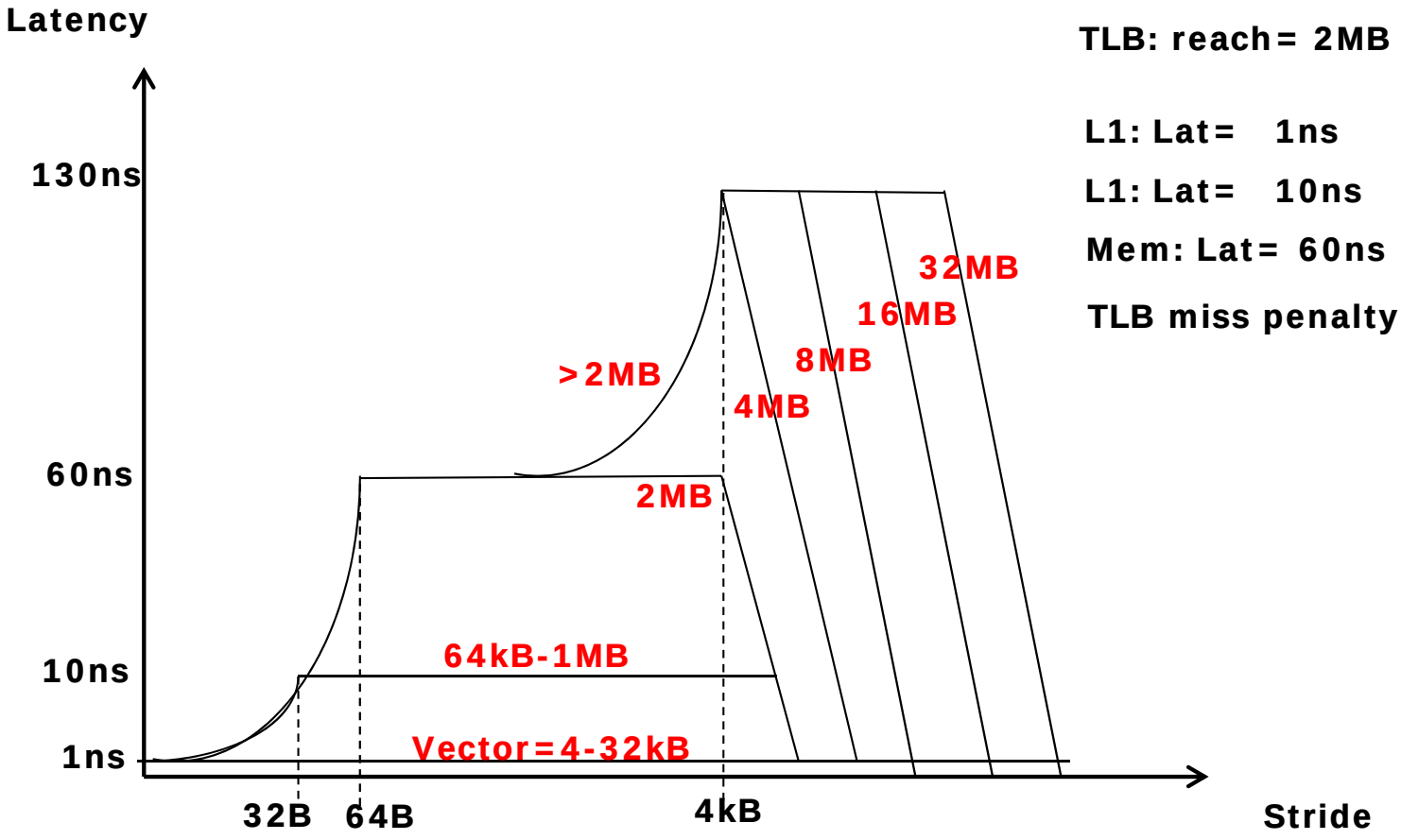
TLB: reach= 2MB #Entries= 512

L1: Lat= 1ns

L1: Lat= 10ns

Mem: Lat= 60ns

TLB miss penalty: 70ns

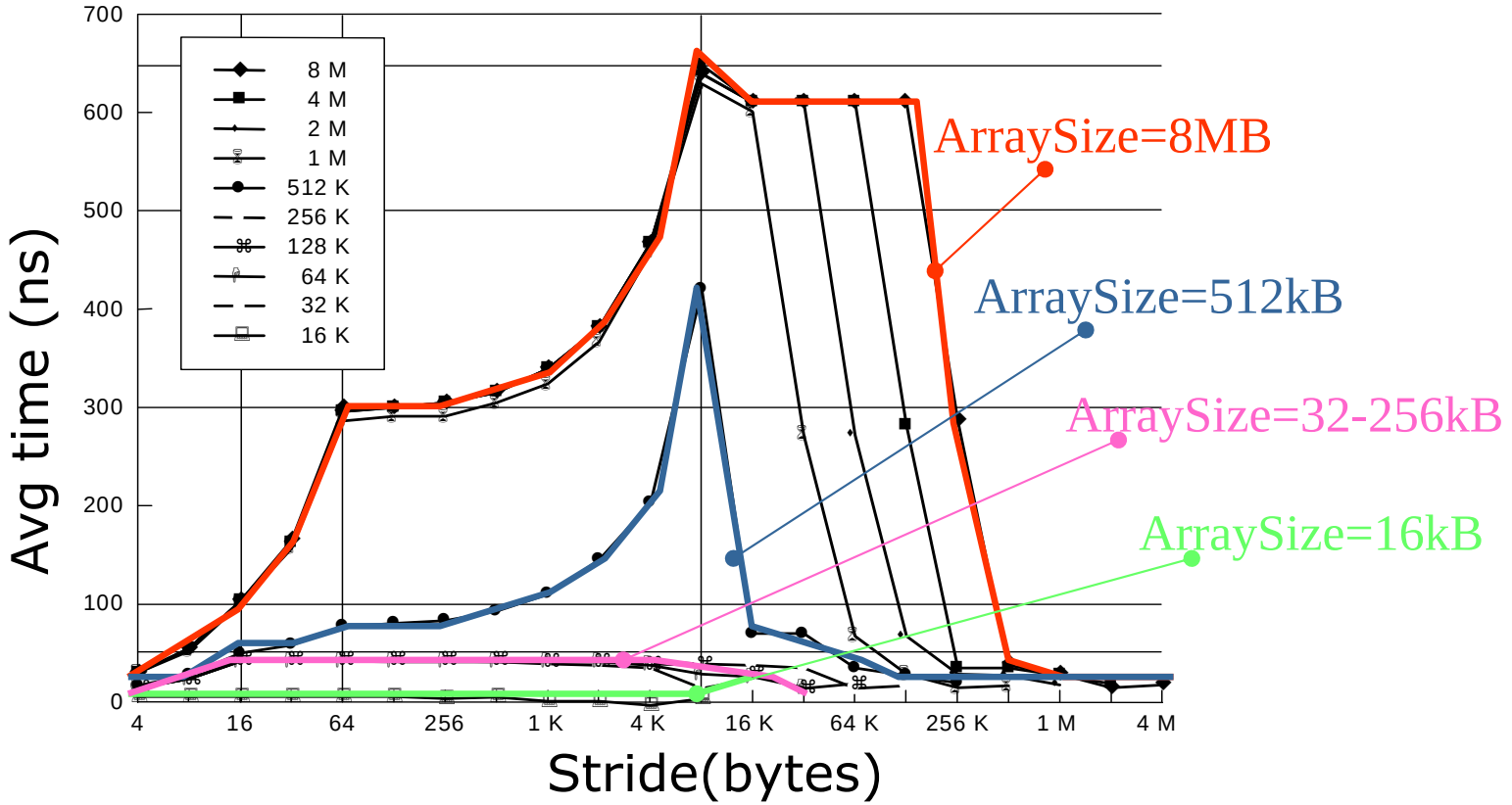


UPPSALA UNIVERSITET

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 */
```

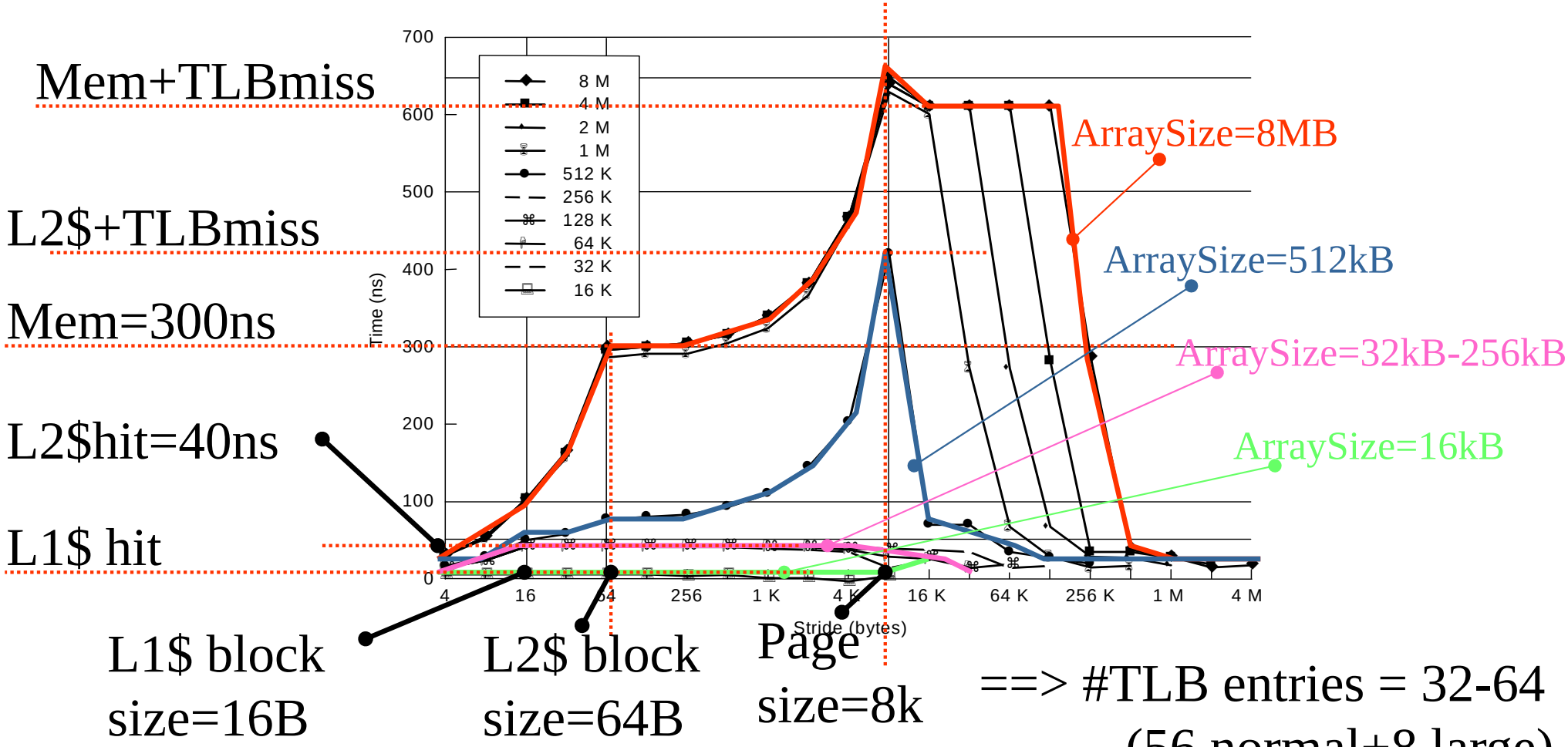


AVDARK 2013

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 */
```

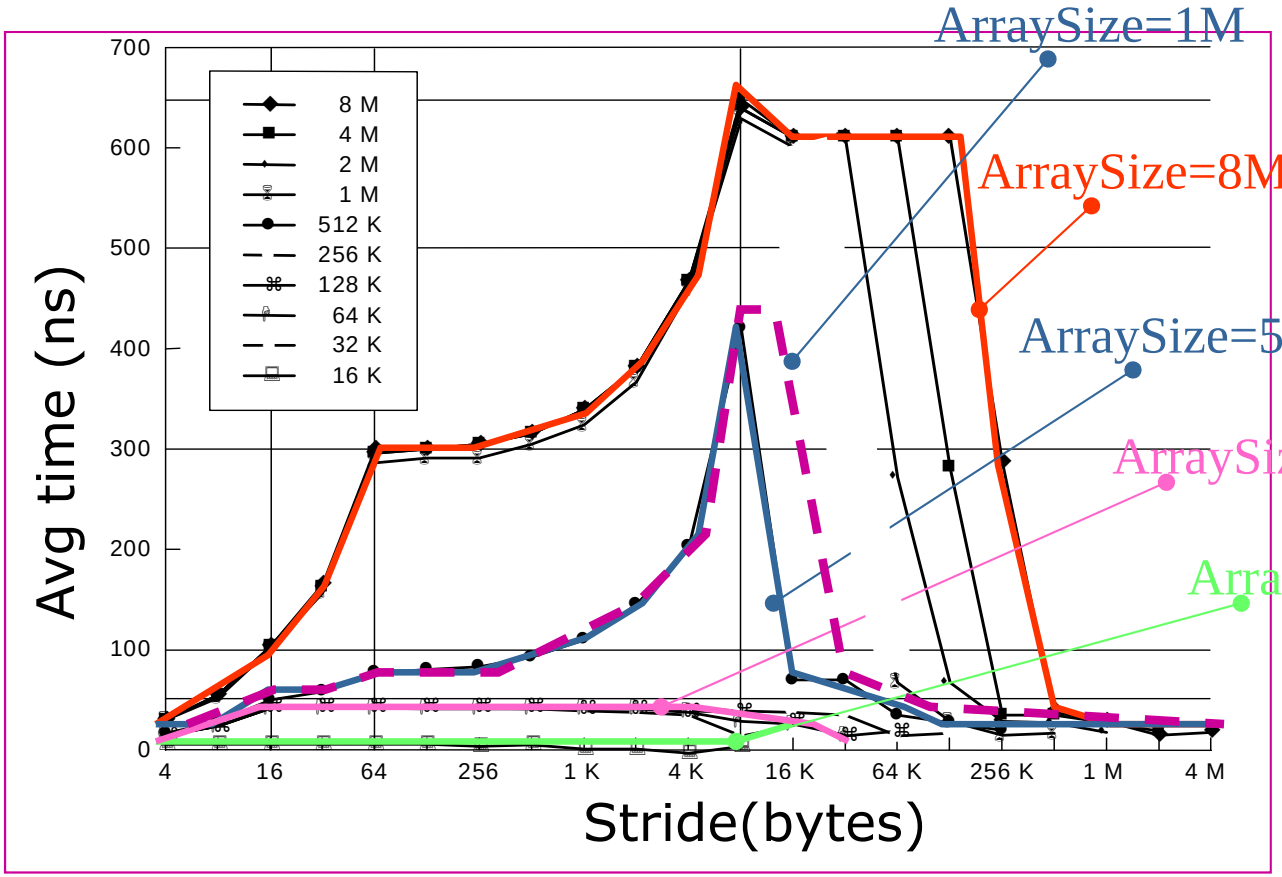


UPPSALA UNIVERSITET

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 */
```



AVDARK 2013

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 */
```