Exact and Fast L1 Cache Simulation for Embedded Systems

Nobuaki Tojo, Nozomu Togawa,
Masao Yanagisawa, Tatsuo Ohtsuki

Dept. of Computer Science and Engineering, Waseda University

Outline

- Introduction
- Basic algorithm
- Janapsatya’s approach
- Our approaches: Configuration Reduction approach by the Cache Behavior 1 and 2
- Experimental results
- Conclusion

Introduction

an embedded system

- a single application
- a class of applications

Its cache configuration can be customized such that an optimal one is achieved

Cache exploration

Generate memory access trace files

$\Downarrow$

Cache evaluation

$\Downarrow$

Output optimal cache configuration

Cache exploration

$$\begin{tikzcd}
\text{Generate memory access trace files} \arrow[d] \\
\boxed{\text{Cache evaluation}} \arrow[d] \\
\text{Output optimal cache configuration}
\end{tikzcd}$$

Existing works

Analytical approach[9]

They have significant errors
Running time is very fast

Simulation based approach[2][6]

no errors
more running time

[2]：J. Edler et al. ,http://www.cs.wise.edu/markhill/DineroIV/
[6]：A.Janapsatya et al. ,in Proc. ASP－DAC, 2006.
[9]：J. J. Pieper et al. ,in Proc. DAC, 2004.

Existing works

Analytical approach[9]

They have significant errors
Running time is very fast

Simulation based approach[2][6]

no errors
more running time

[2]：J. Edler et al. ,http://www.cs.wise.edu/markhill/DineroIV/
[6]：A.Janapsatya et al. ,in Proc. ASP－DAC, 2006.
[9]：J. J. Pieper et al. ,in Proc. DAC, 2004.

Simulation based approach

Janapsatya's approach [6]

In simulation based approaches,
this is the fastest algorithm

+ New cache properties

Our approach

Configuration Reduction approach by the Cache Behavior

CRCB1 and CRCB2（Exact Simulation）

An average of 1.8x faster than Janapsatya's approach [6]

[6]：A.Janapsatya et al. , in Proc. ASP-DAC, 2006.

Cache design space

Cache configuration<t,b,a>

Cache size	$t = 2^i : 6 \le i \le 22$
Line size	$b = 2^i : 3 \le i \le 10$
Associativity	$a = 2^i : 0 \le i \le 4$

Totally 460 cache configurations

Total memory access time

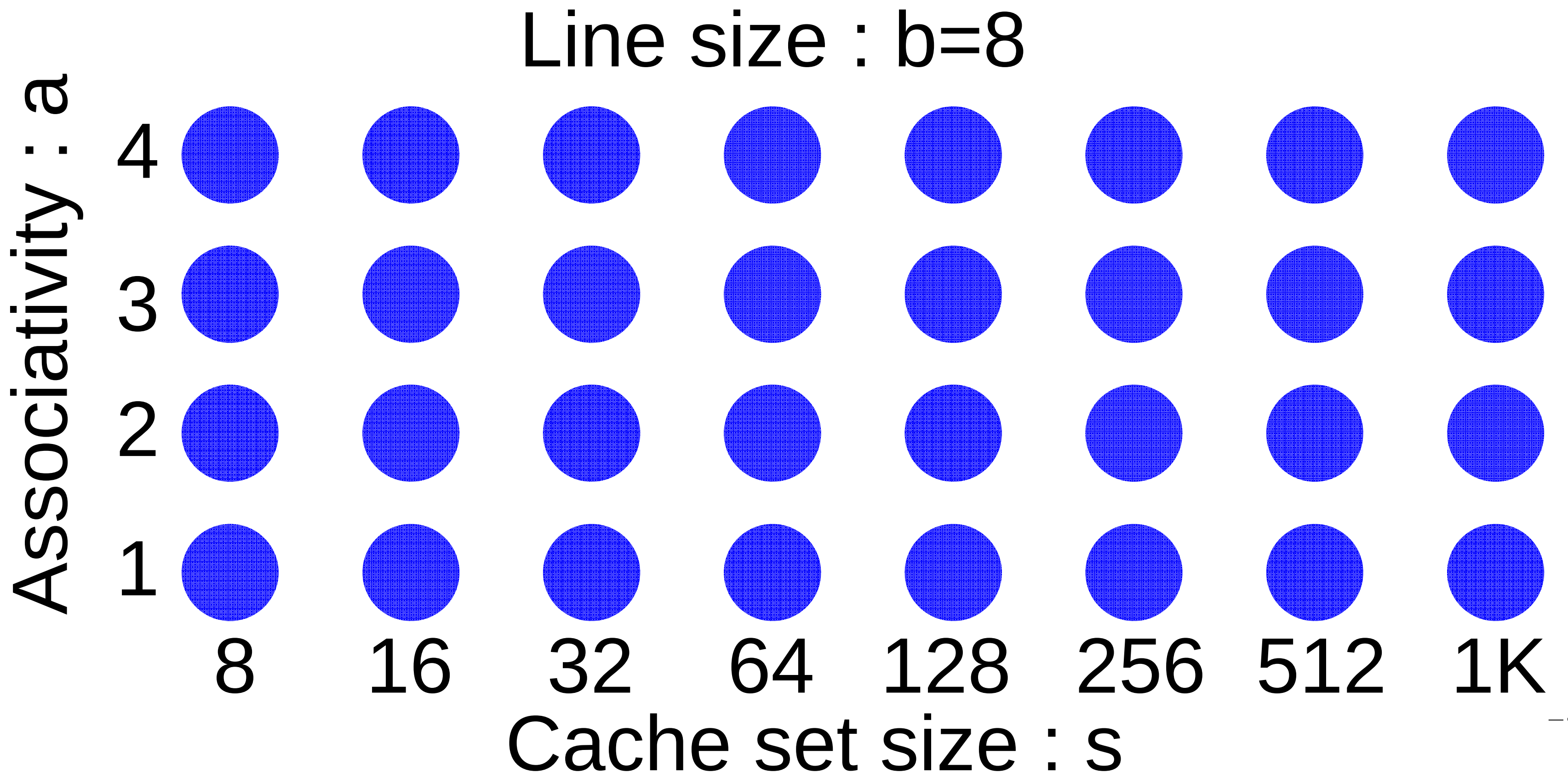
- Total memory access time:T
  - Cache access time:T_c
  - Number of memory access:n
  - Main memory access time:T_m ＝19.5[ns]
  - Memory transfer time:$T_{c\text{-}m}$ ＝20[B/ns]
  - The number of cache misses:n_{miss}

$$T = T_c \times n + \left(T_m + b \times T_{c-m}\right) \times n_{miss}$$

We minimize the total memory access time

Basic algorithm

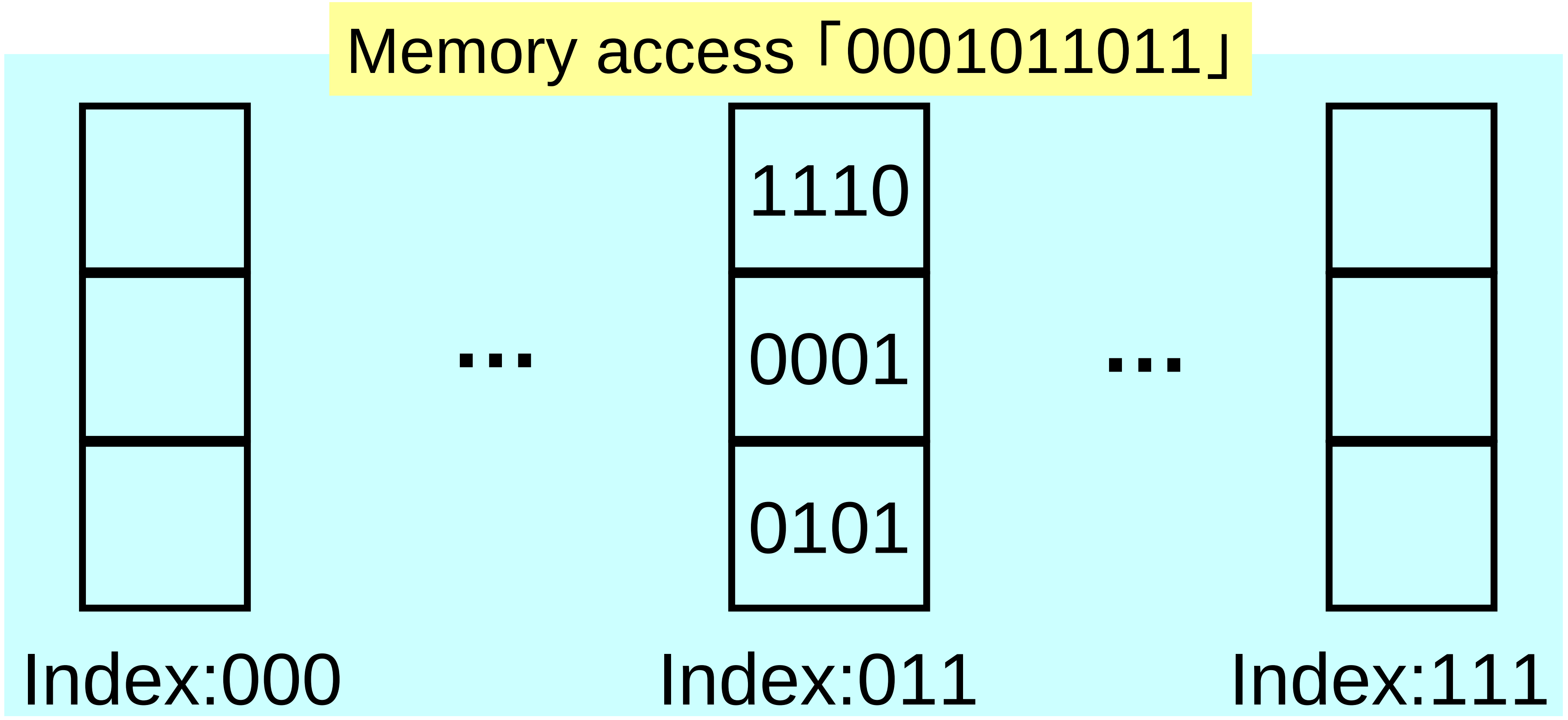
Simulate all cache configurations
Least Recent Used (LRU) replacement



Basic algorithm

Example：

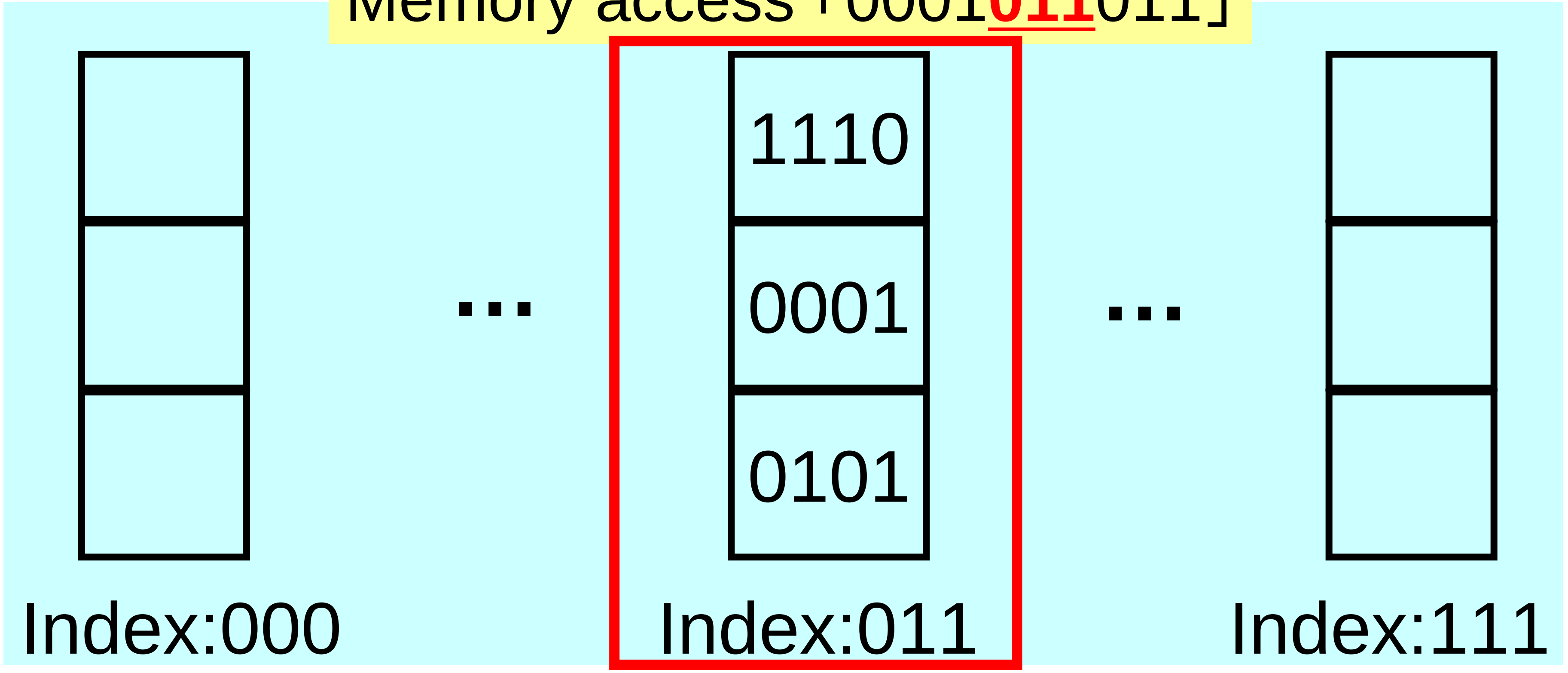
cache set size s = 8，associativity a=3，line size b=8



Basic algorithm

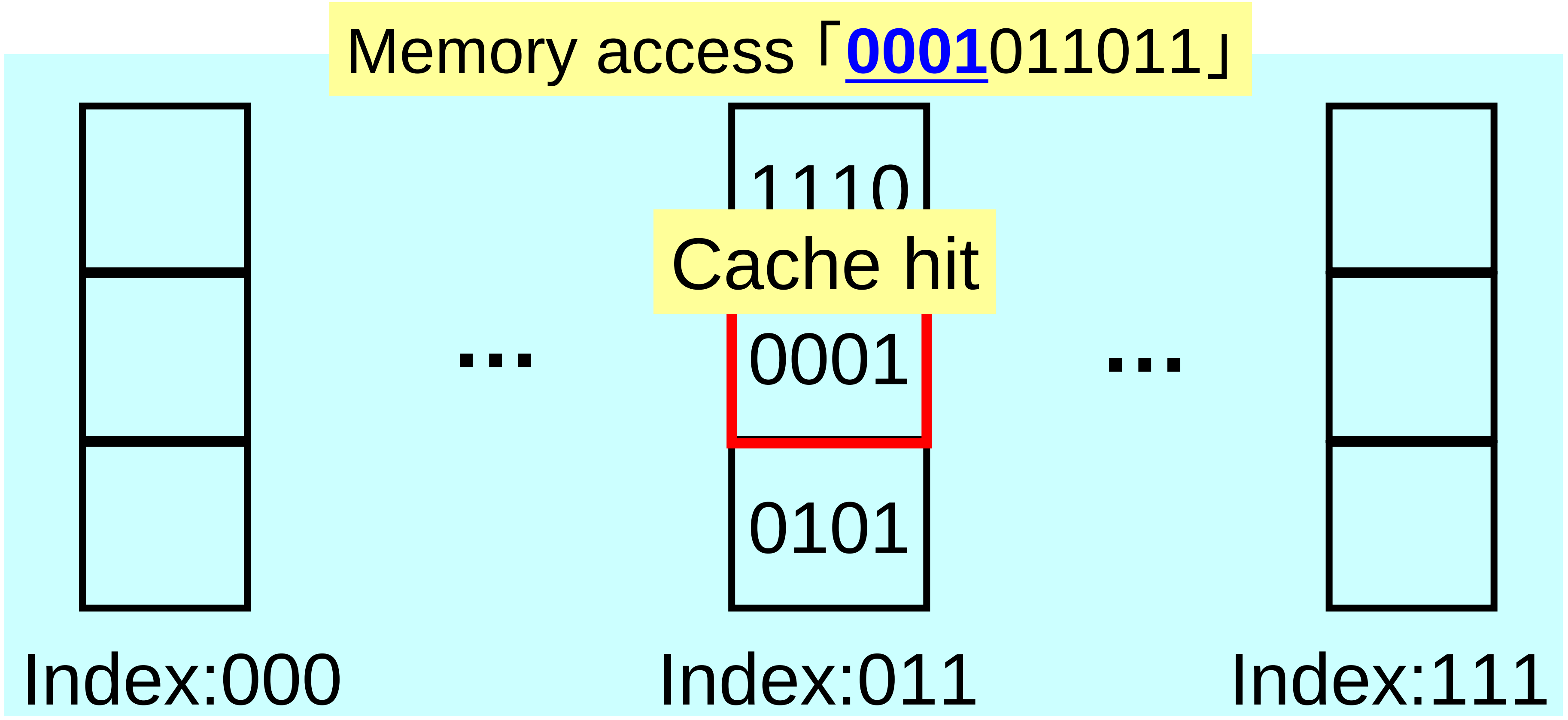
Example：
cache set size s = 8，associativity a=3，line size b=8

Memory access「0001**011**011」



Basic algorithm

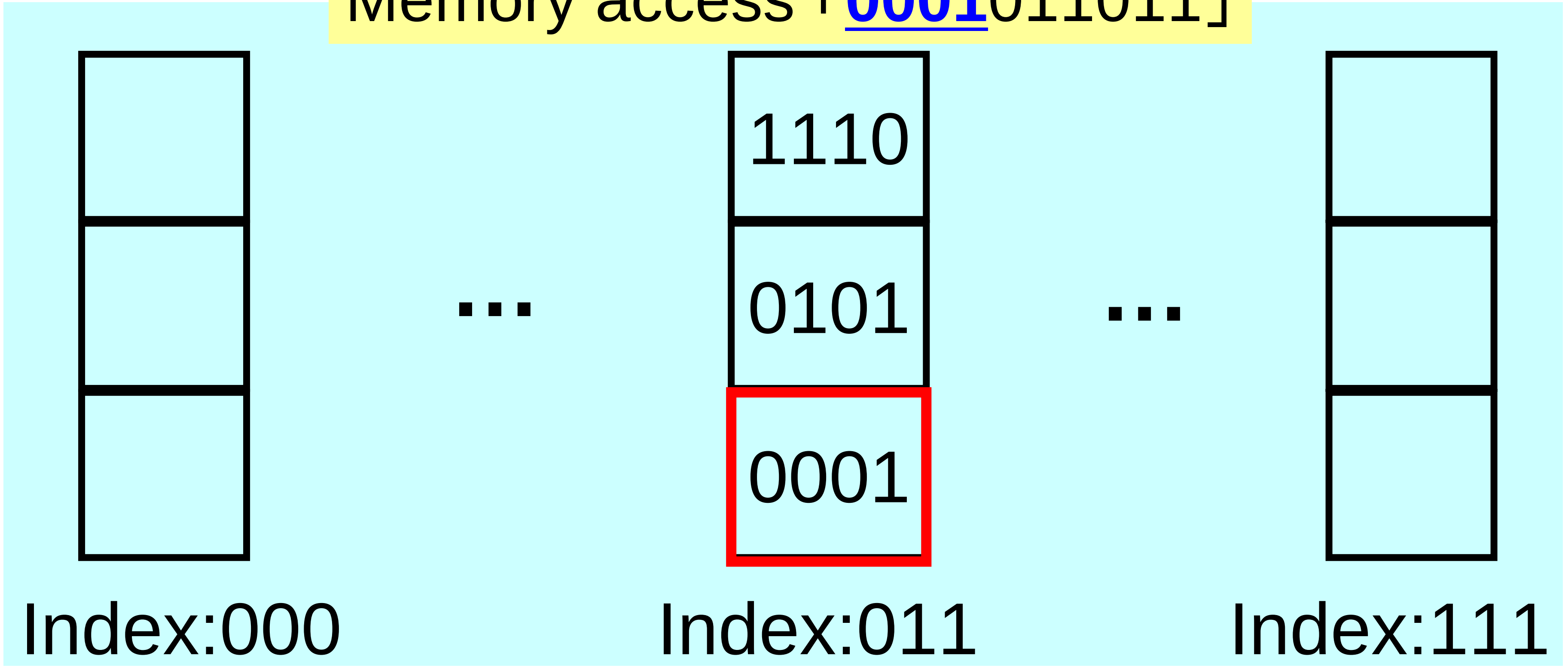
Example：
cache set size s = 8，associativity a=3，line size b=8



Basic algorithm

Example：
cache set size s = 8，associativity a=3，line size b=8

Memory access「**0001**011011」

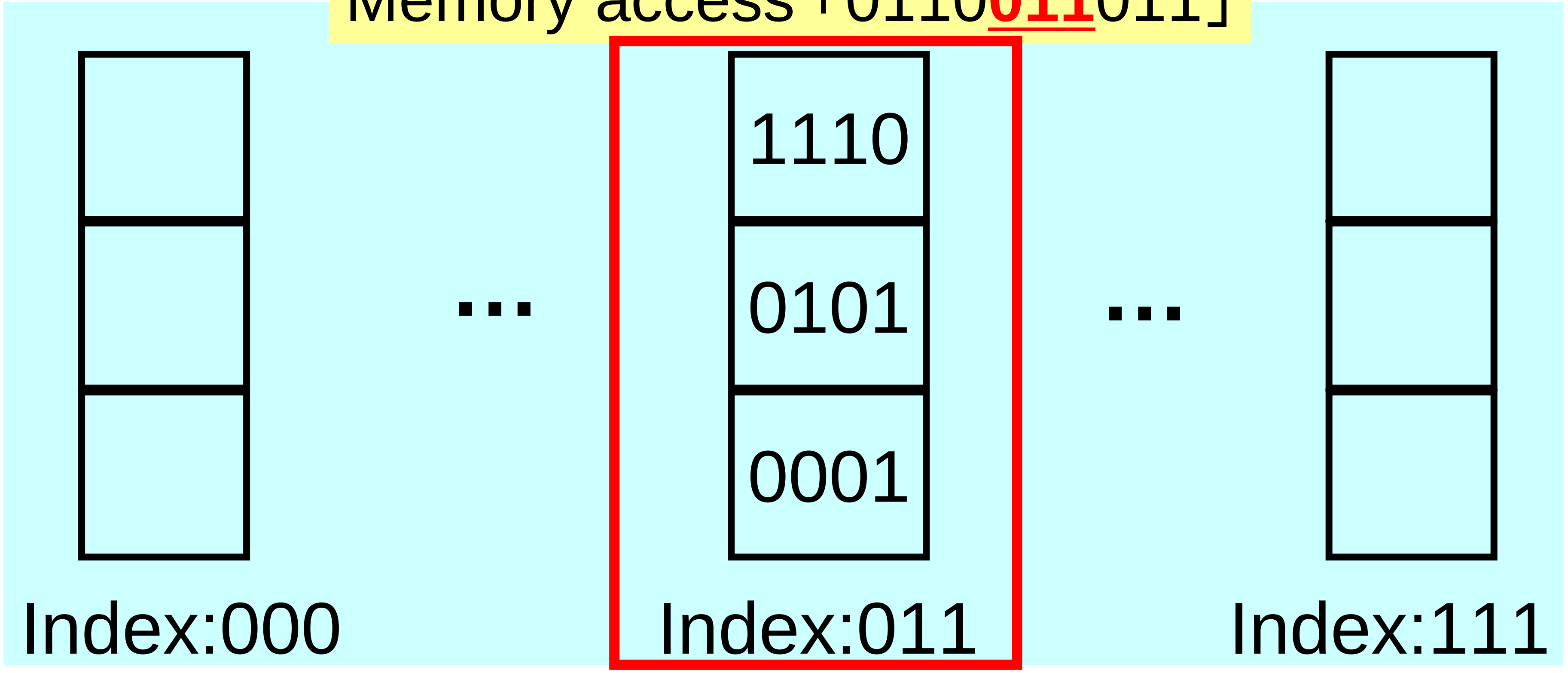


Basic algorithm

Example：

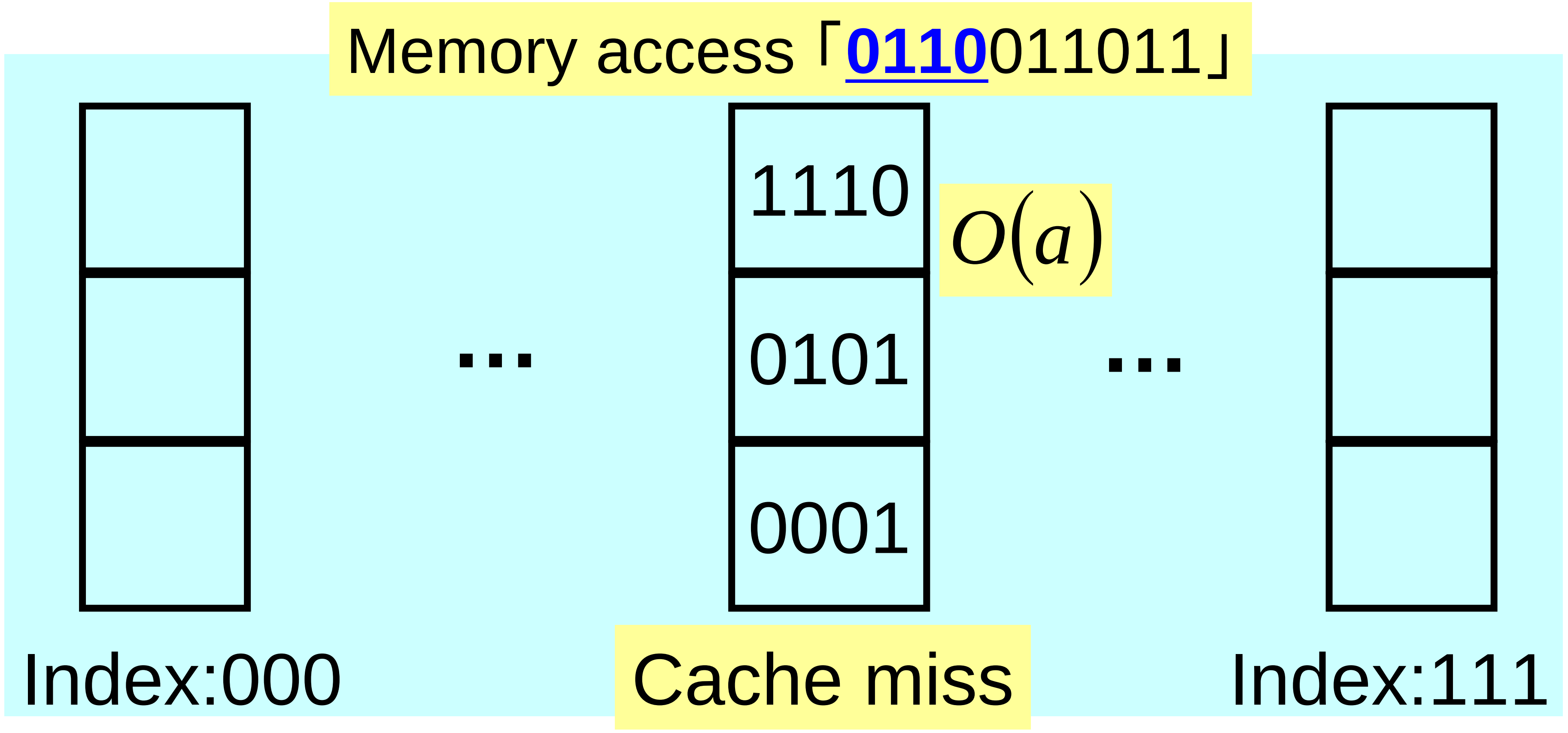
cache set size s = 8, associativity a=3, line size b=8

Memory access「0110**011**011」



Basic algorithm

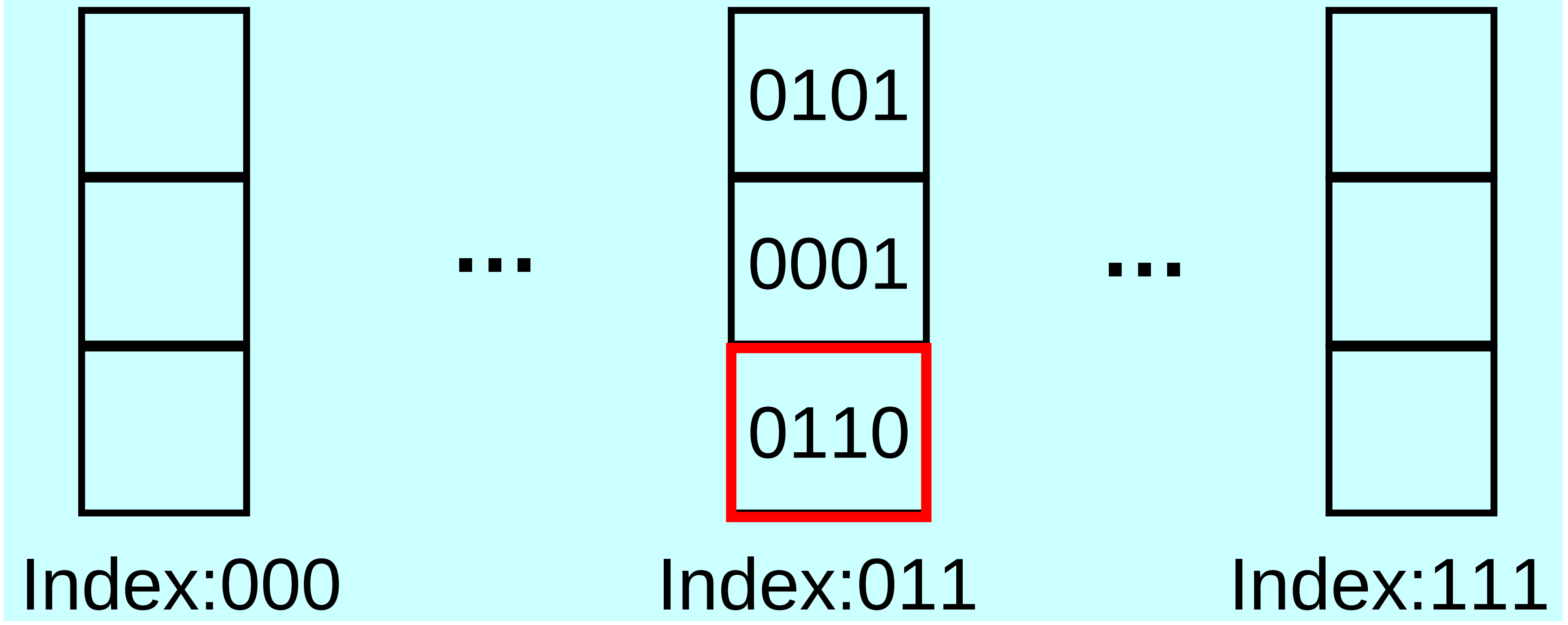
Example：
cache set size s = 8，associativity a=3，line size b=8



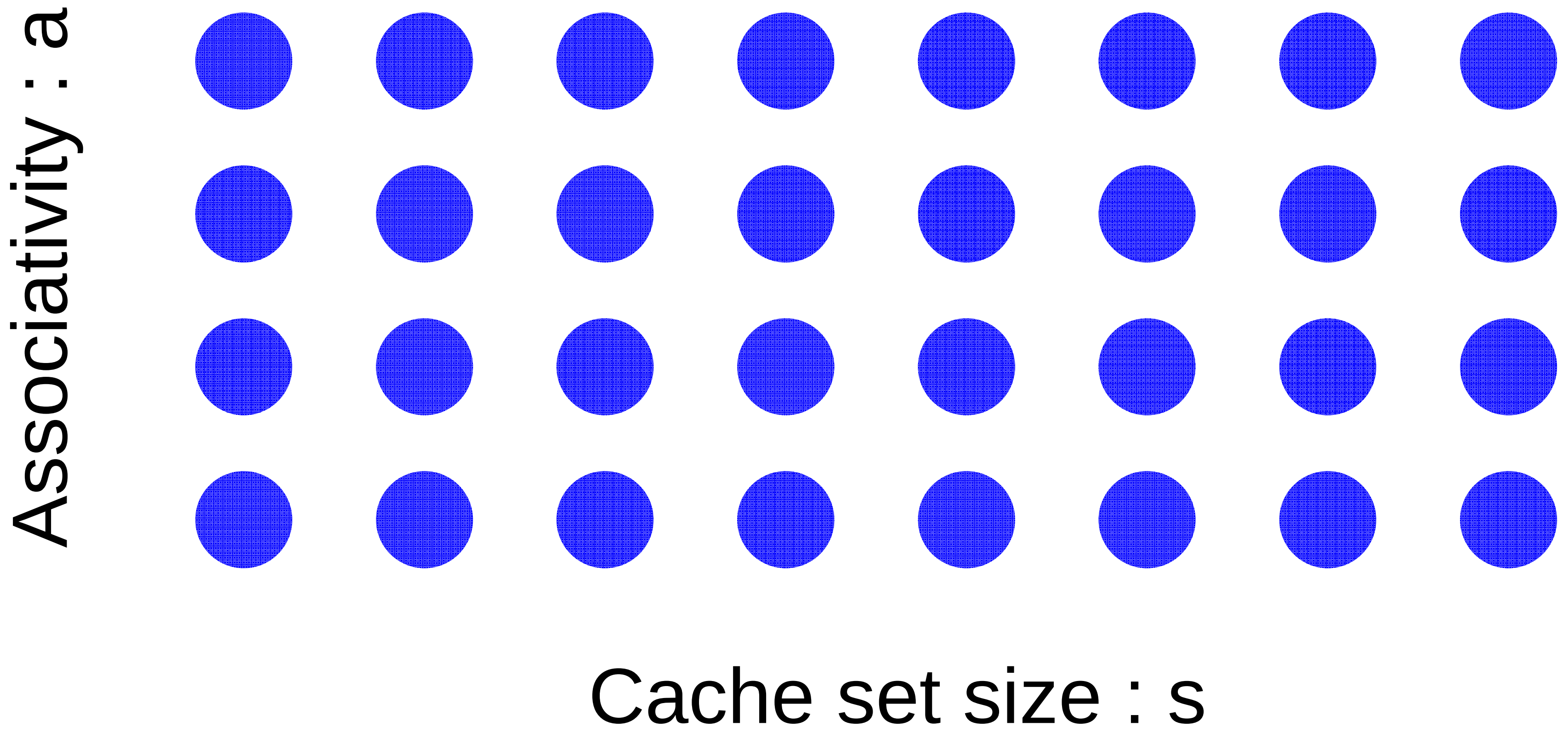
Basic algorithm

Example：
cache set size s = 8，associativity a=3，line size b=8

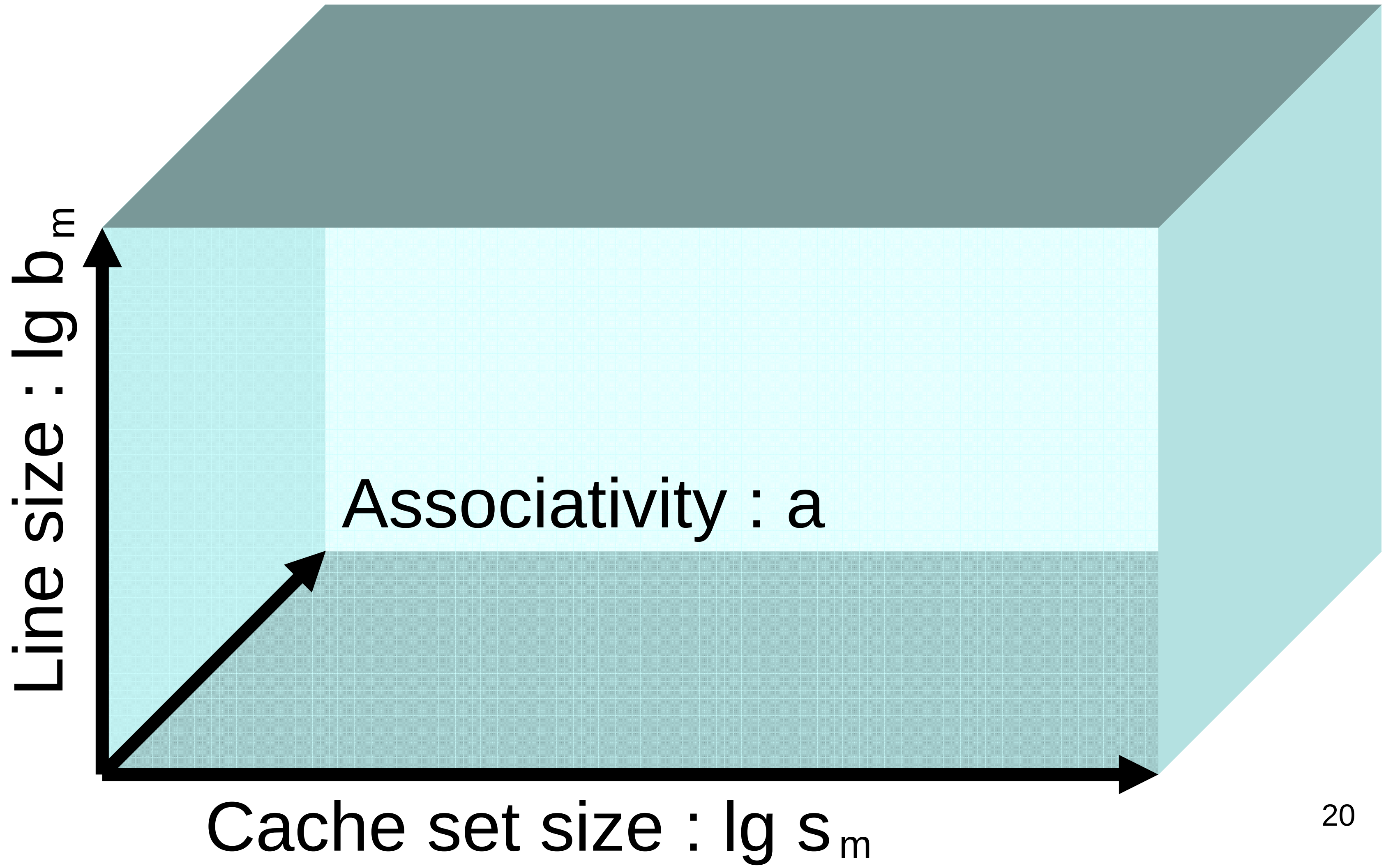
Memory access「**0110**011011」



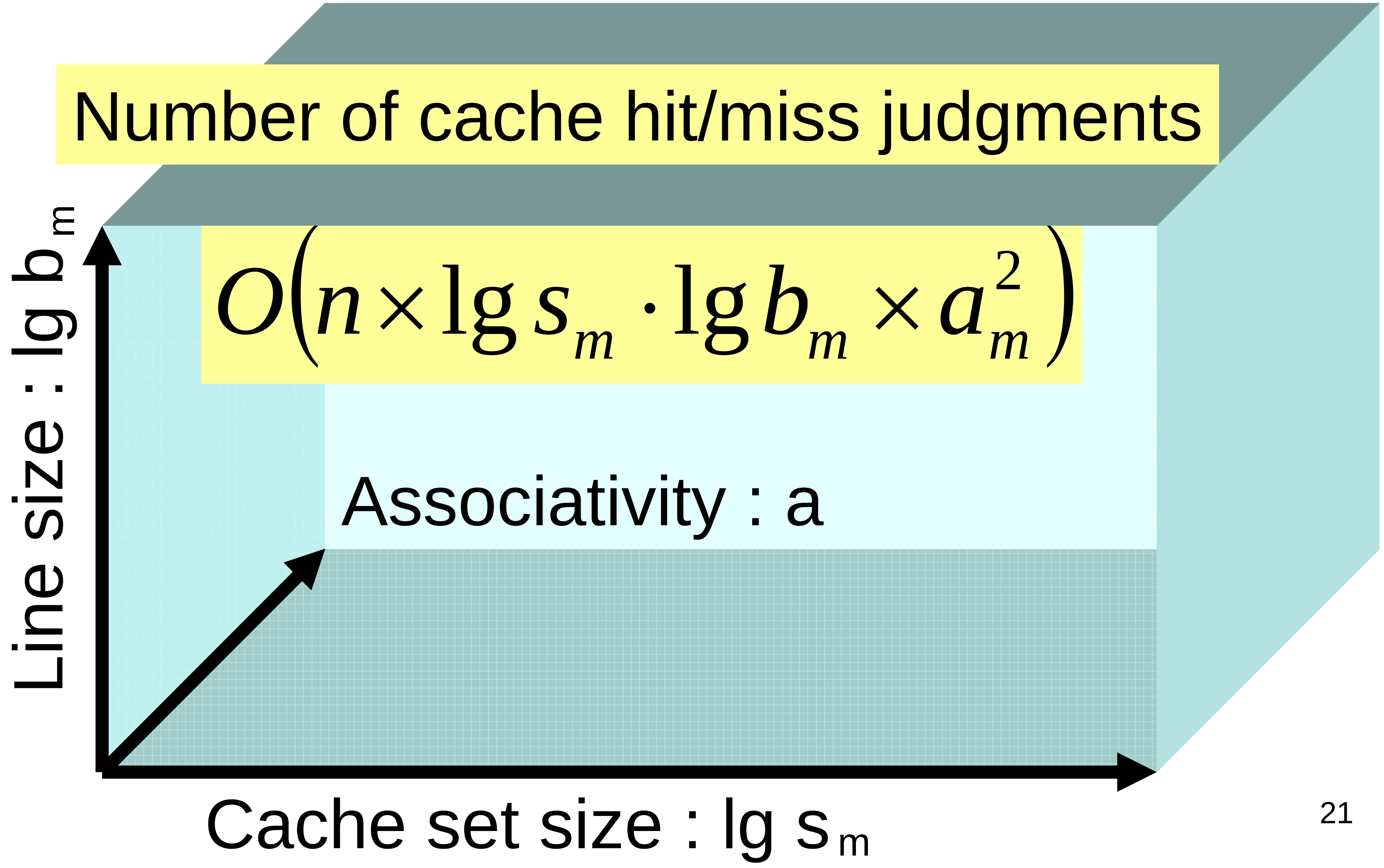
Basic algorithm



Basic algorithm



Basic algorithm


Number of cache hit/miss judgments
$O\left(n \times \lg s_m \cdot \lg b_m \times a_m^2\right)$
Associativity : a
Line size : lg b_m
Cache set size : lg s_m

Fast simulation

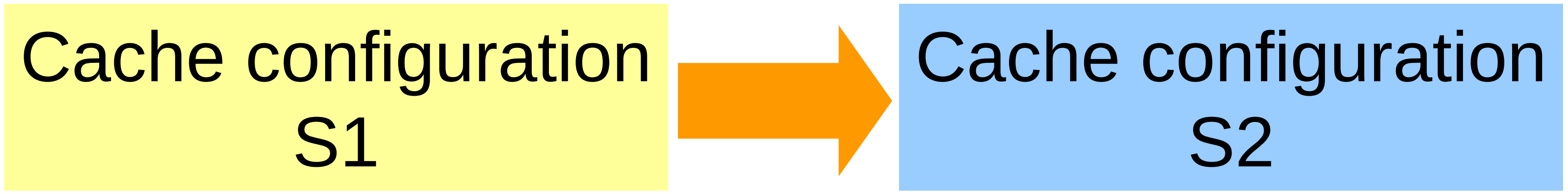
Decrease
the number of cache hit/miss judgments

Skipped cache simulations

Janapsatya's approach[6],CRCB1, and CRCB2

[6]：A.Janapsatya et al. , in Proc. ASP-DAC, 2006.

Inclusion Property



All the contents of S1 are subset of the contents of S2

Inclusion Property

$$\begin{tikzcd}\text{Cache configuration S1} \arrow[r] & \text{Cache configuration S2}\end{tikzcd}$$

All the contents of S1 are subset of the contents of S2

Inclusion Property [8]
If a cache miss occurs in S2,
it also occurs in S1
If a cache hit occurs in S1,
it also occurs in S2

[8]：R. L. Mattson et al. ,IBM System Journal, 1970.

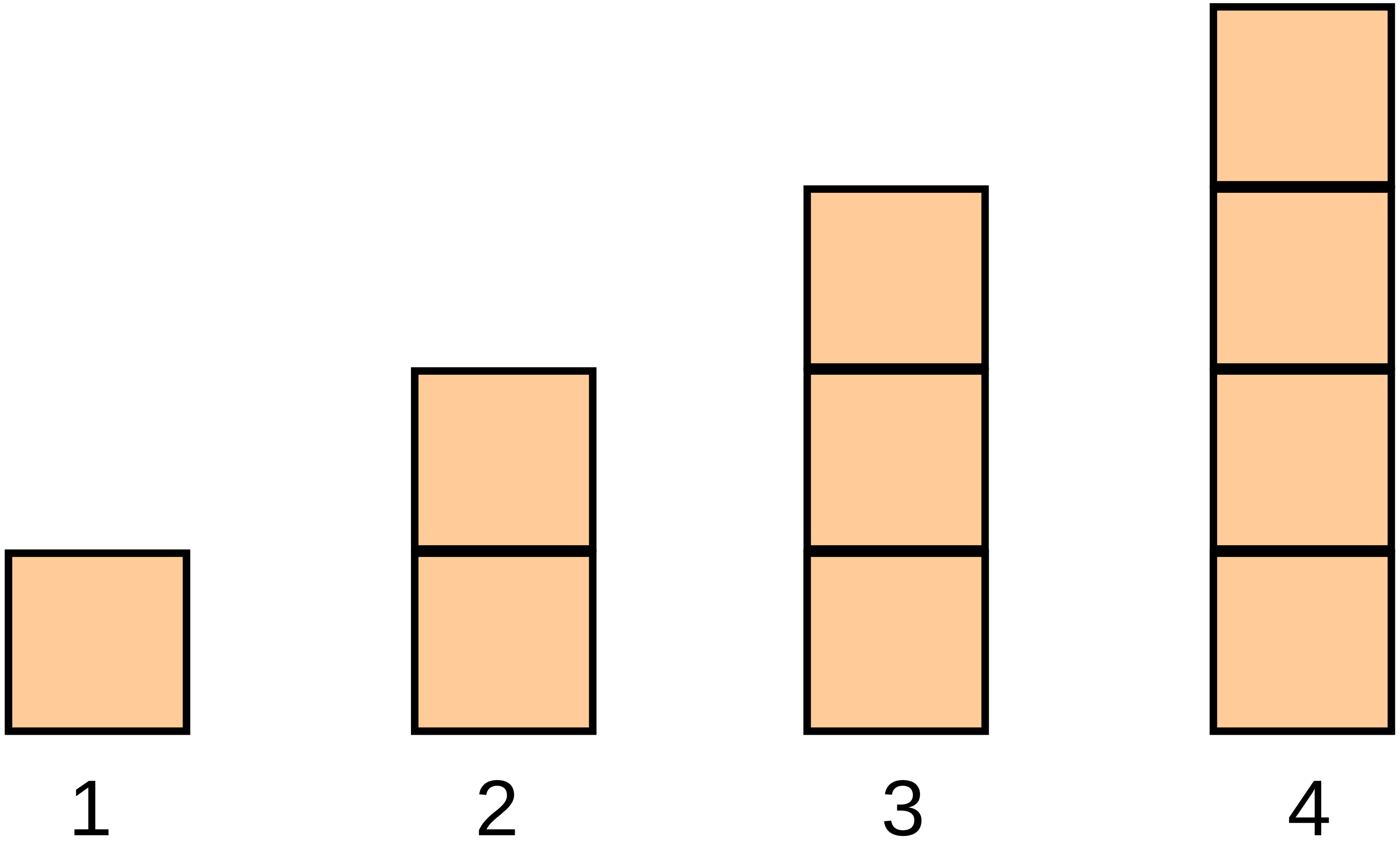
Simulation based approaches

- Existing works
  - Apply the Inclusion Property to an associativity
    - Janapsatya's approach [1]
- Our approaches
  - Apply the Inclusion Property to a cache set size
    - CRCB1
  - Apply the Inclusion Property to a line size
    - CRCB2

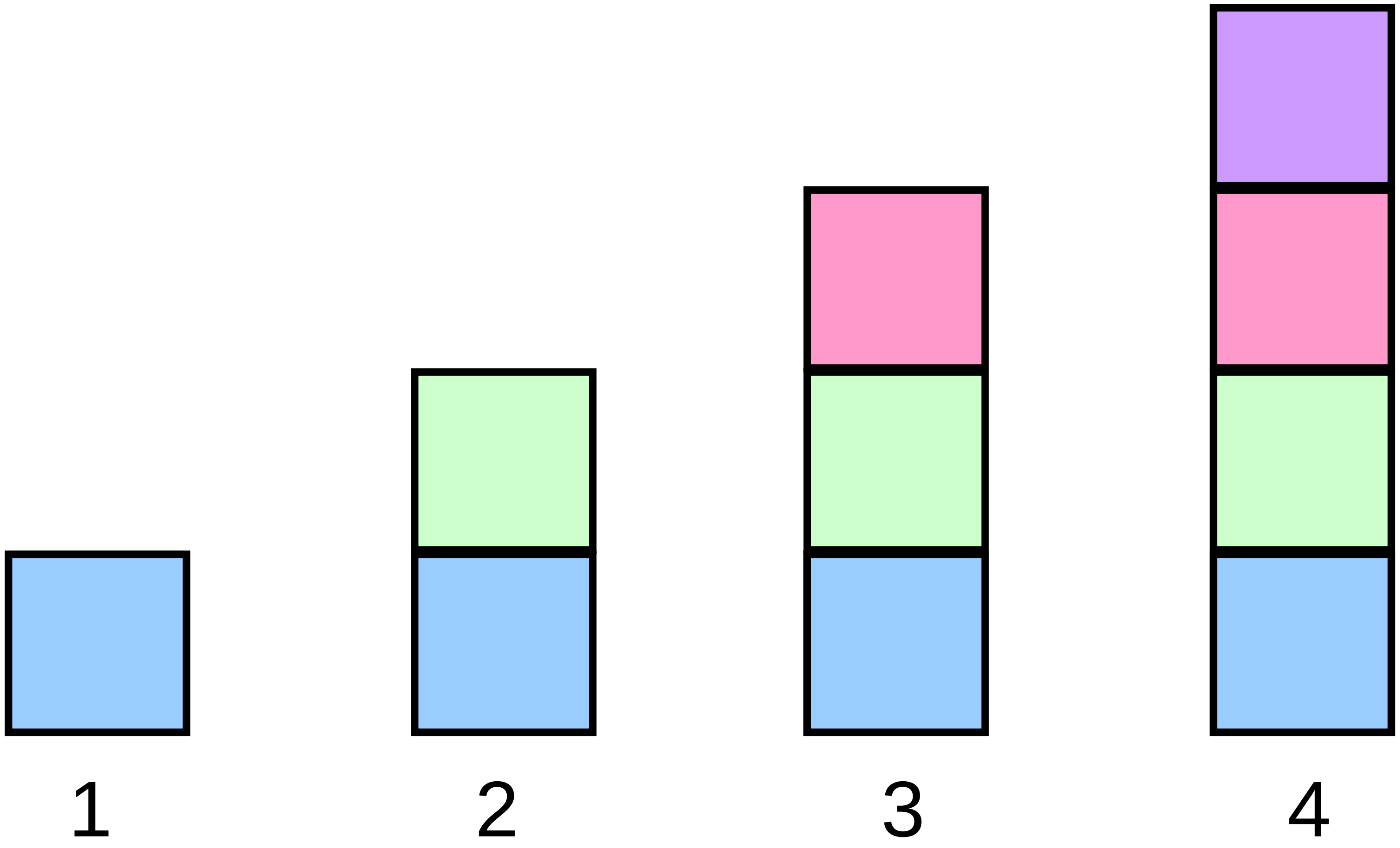
All approaches are exact simulations

[6]：A.Janapsatya et al. , in Proc. ASP-DAC, 2006.

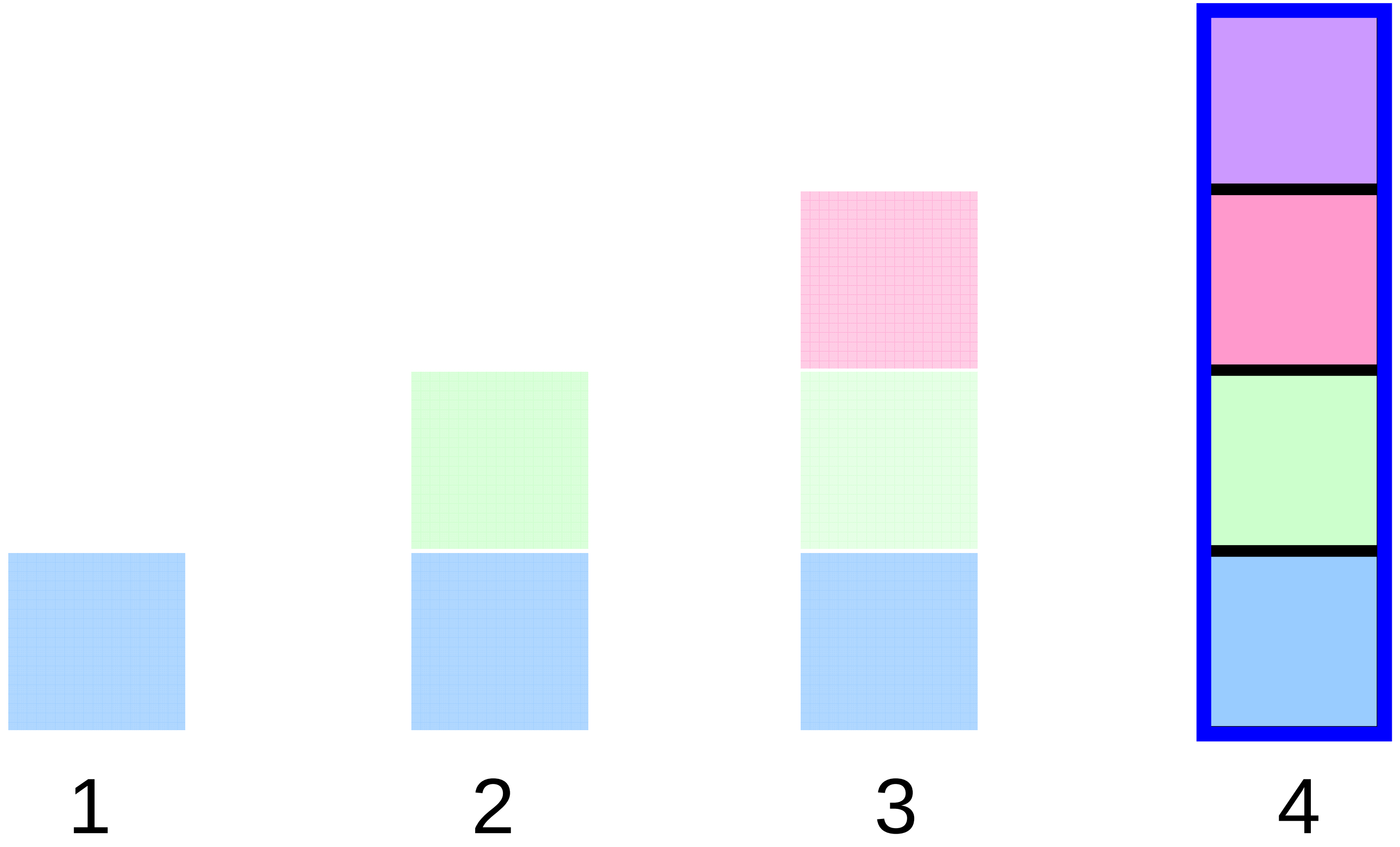
Janapsatya's approach [6]



Janapsatya’s approach [6]



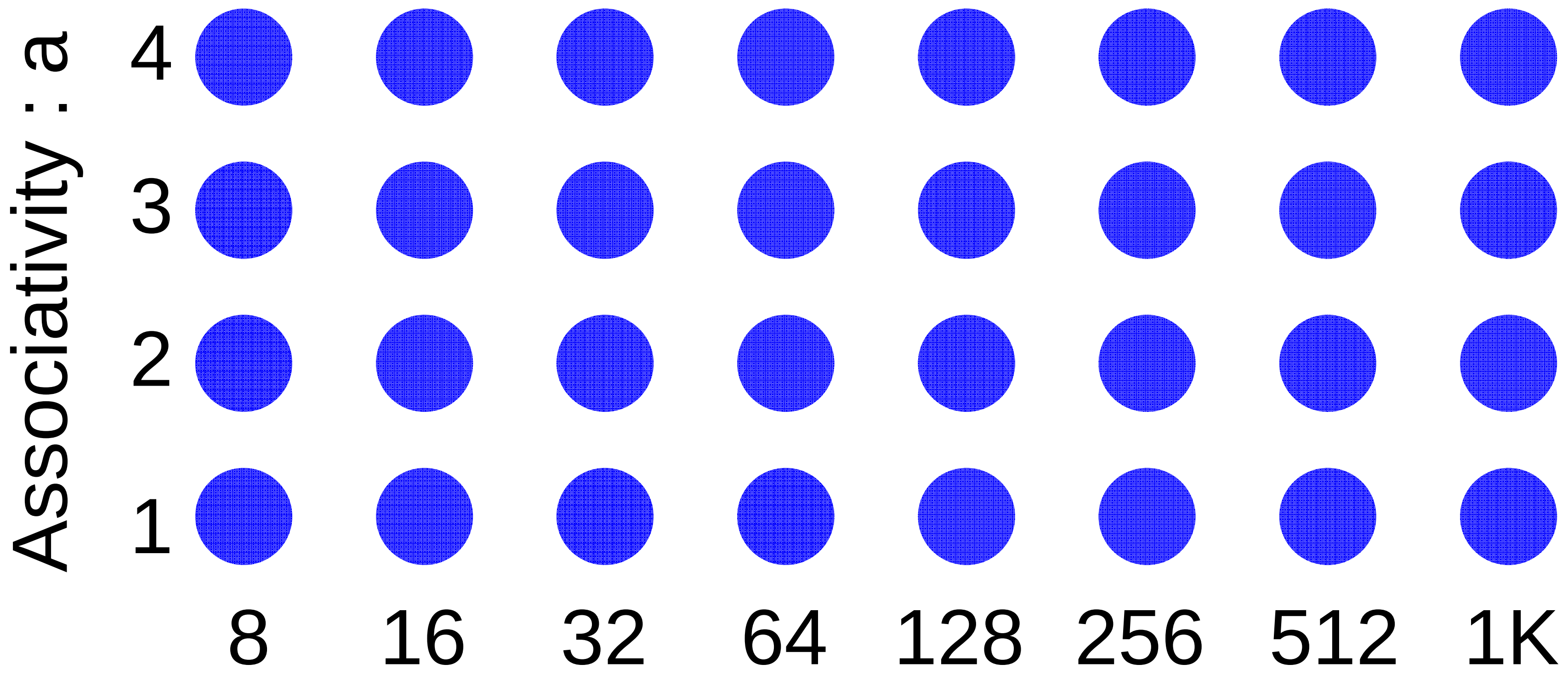
Janapsatya's approach [6]



Janapsatya's approach [6]

We respect to the memory access m_a

The line size : b=k

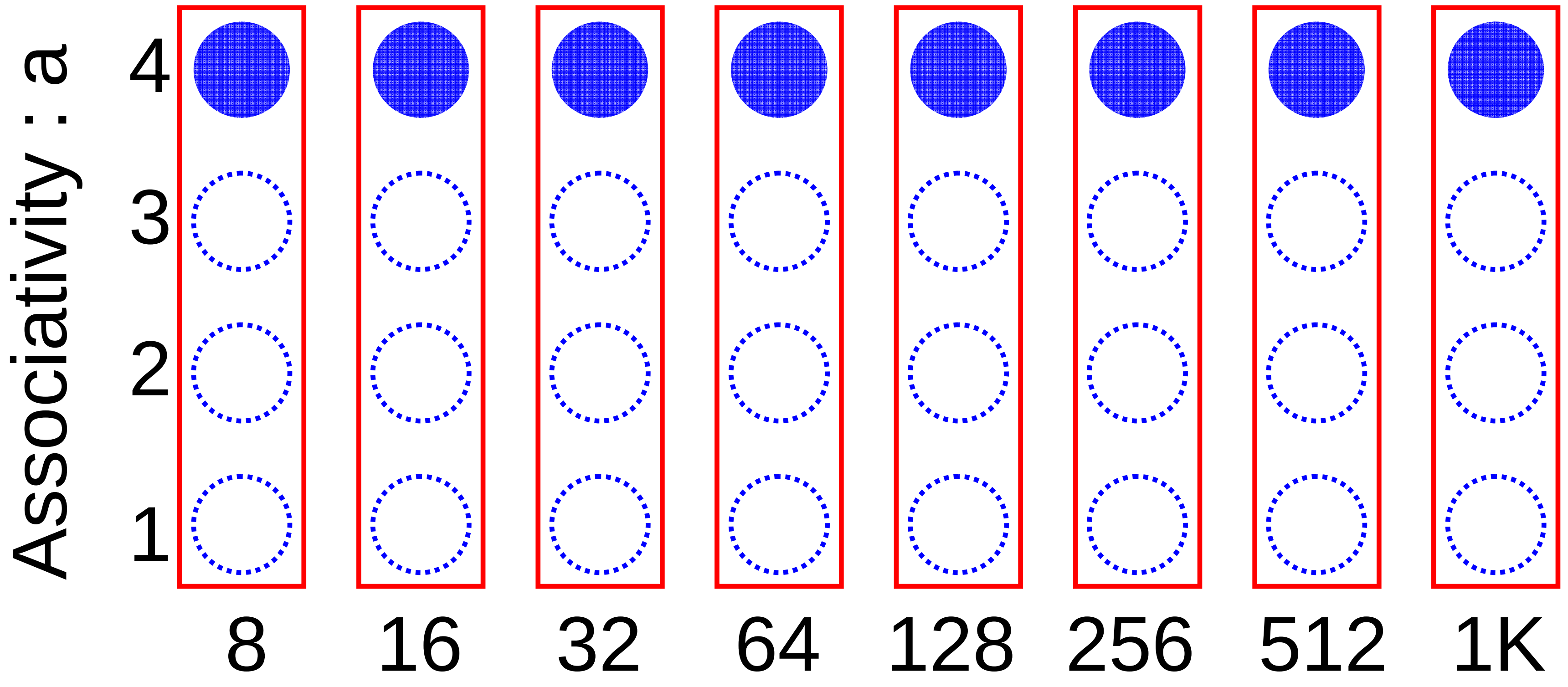


Cache set size : s

Janapsatya's approach [6]

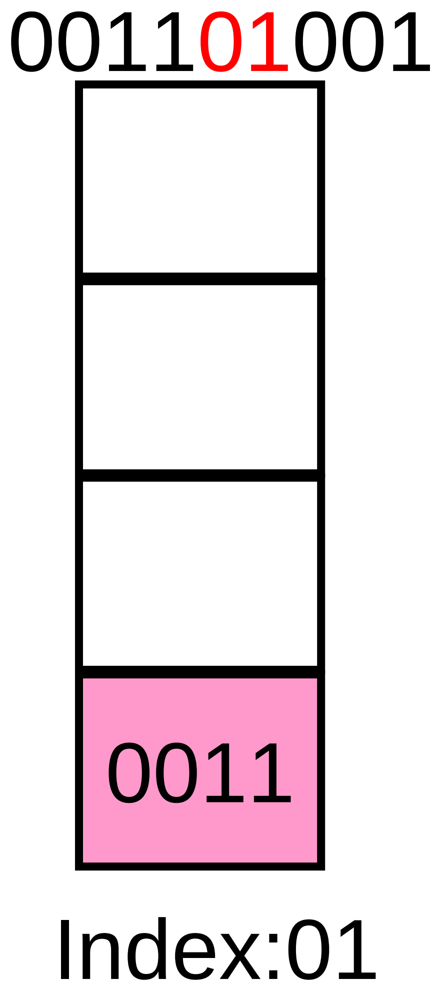
We respect to the memory access m_a

The line size : b=k

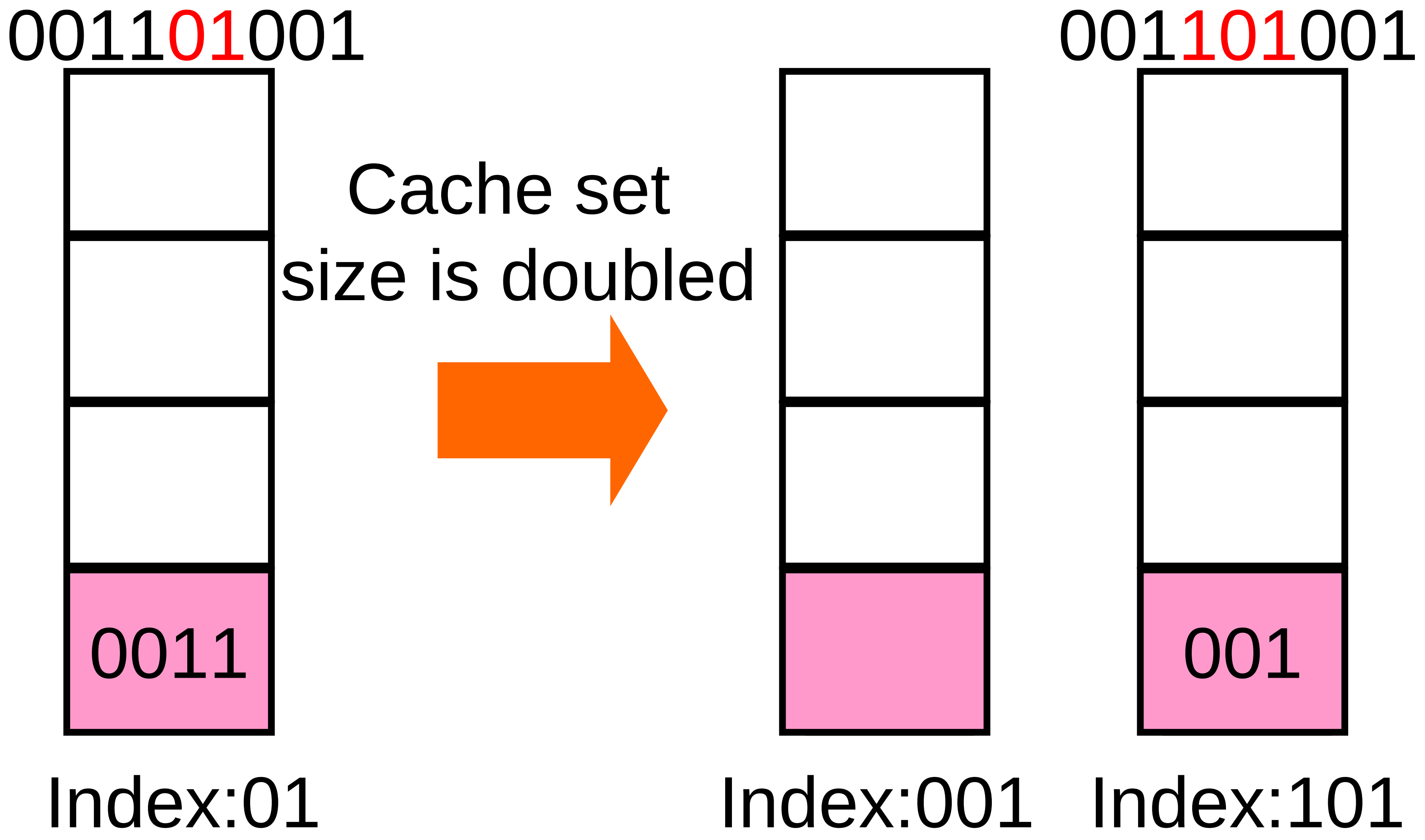


Cache set size : s

The property of CRCB1



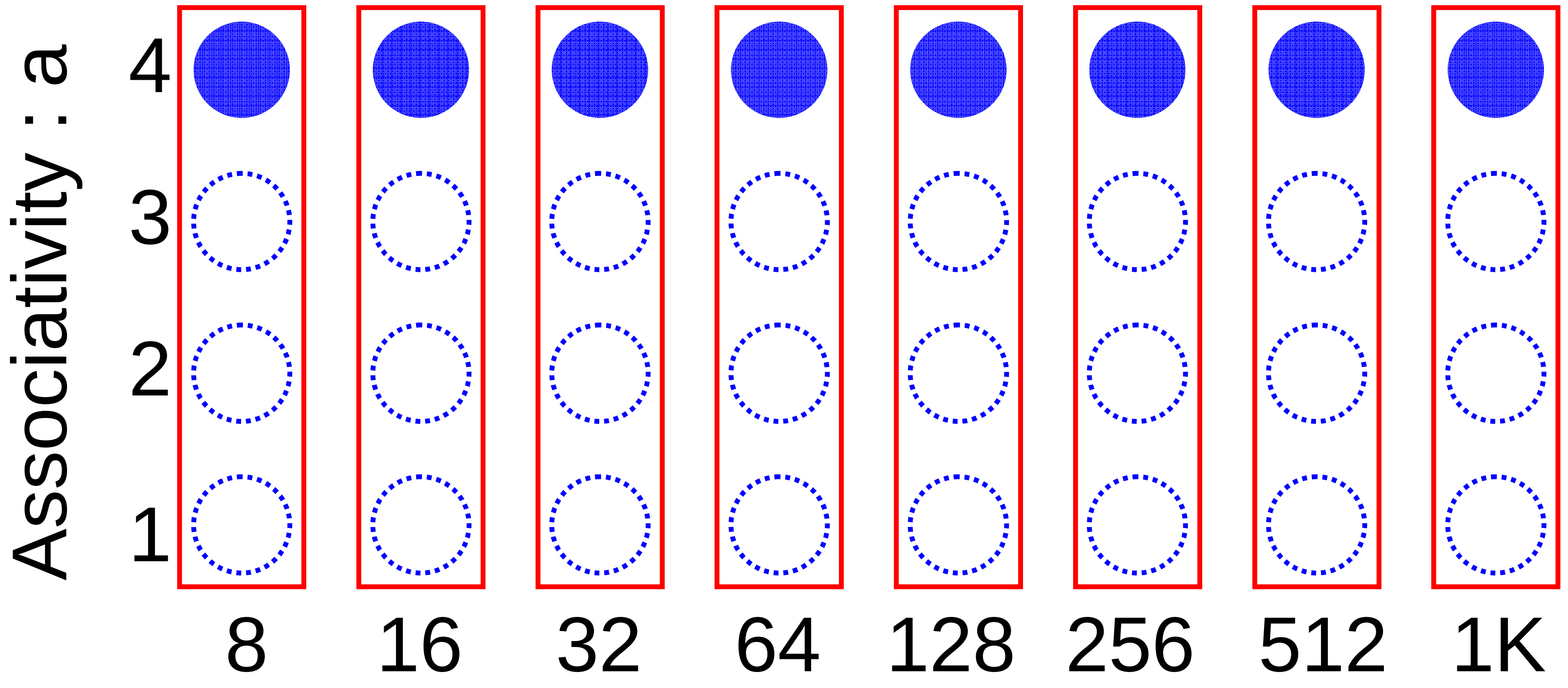
The property of CRCB1



CRCB1

We respect to the memory access m_a

The line size : b=k

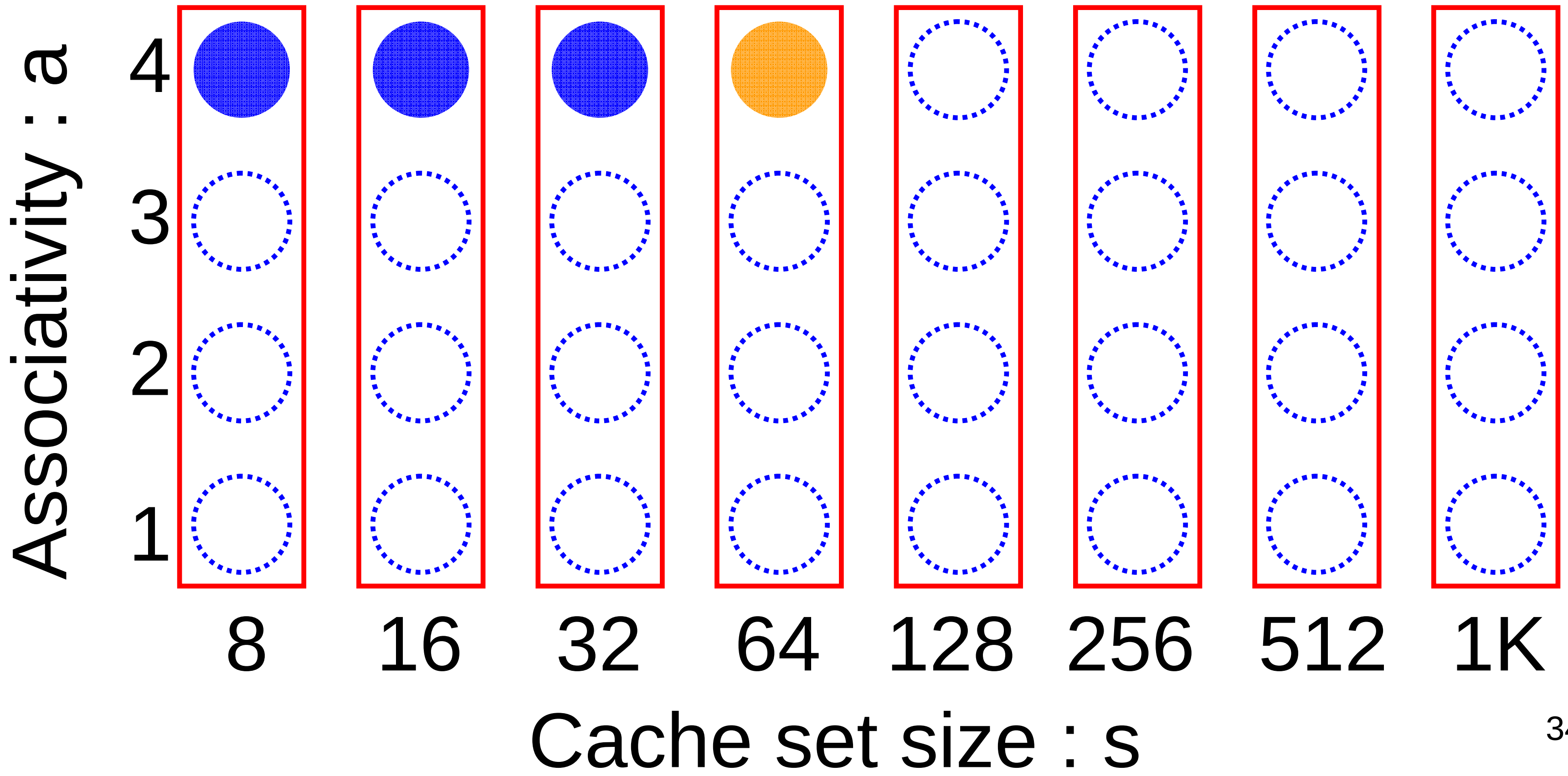


Cache set size : s

CRCB1

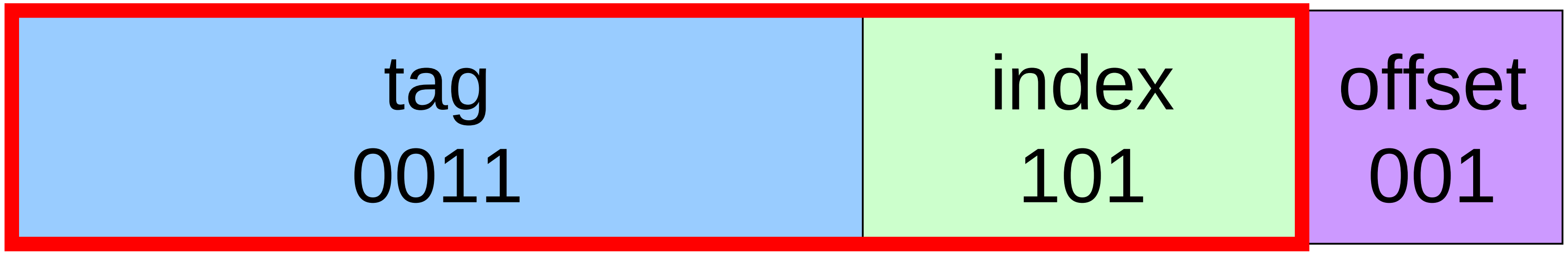
We respect to the memory access m_a

The line size : b=k

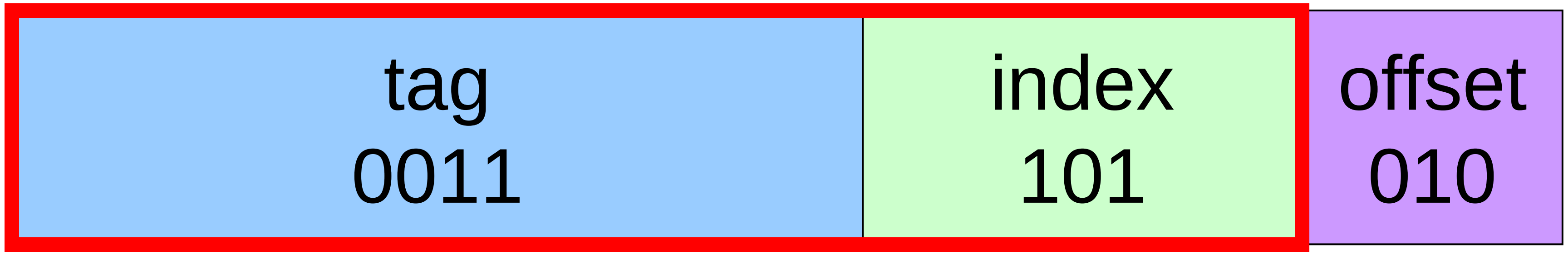


The property of CRCB2

The memory address of (i－1)th memory access

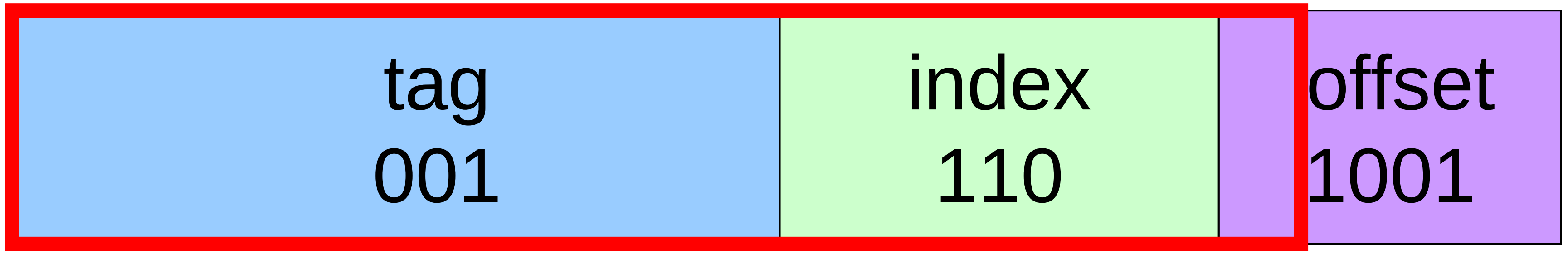


The memory address of i-th memory access

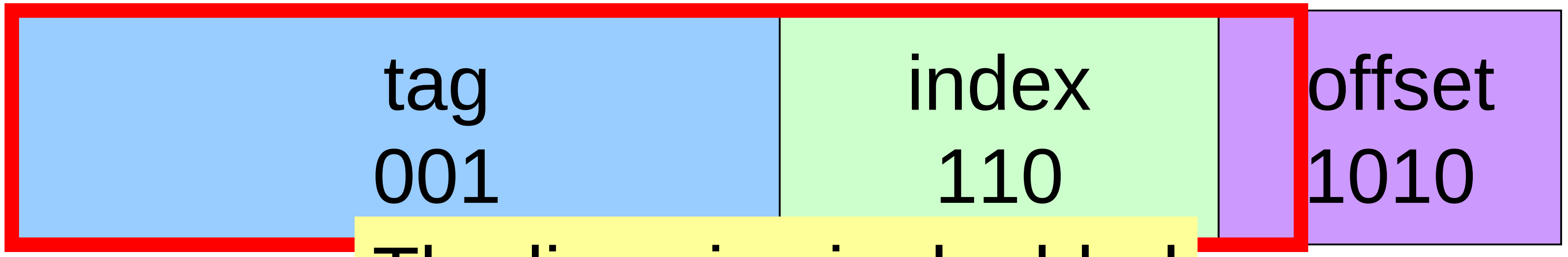


The property of CRCB2

The memory address of (i－1)th memory access



The memory address of i-th memory access

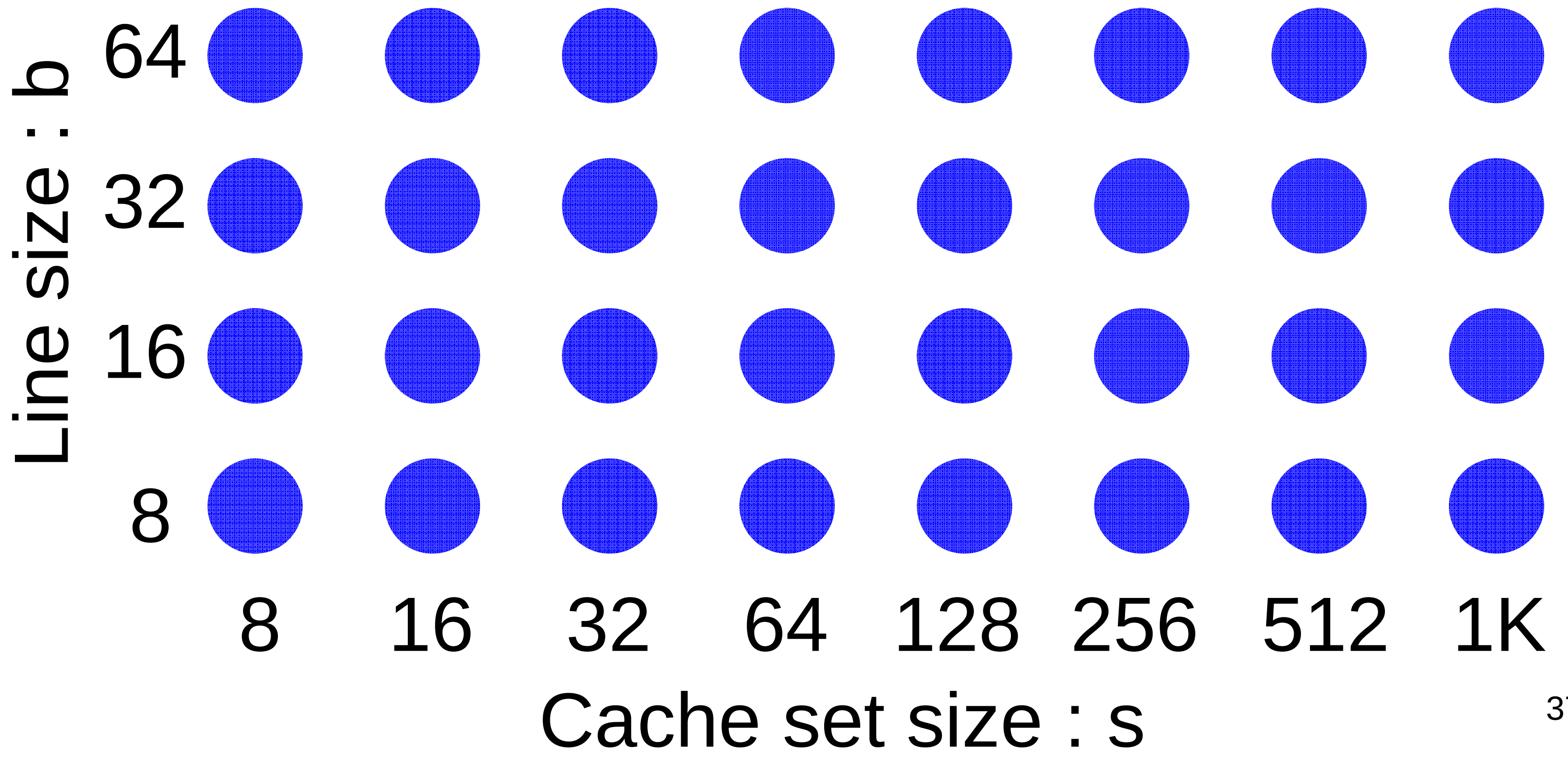


The line size is doubled

CRCB2

We respect to the memory access m_a

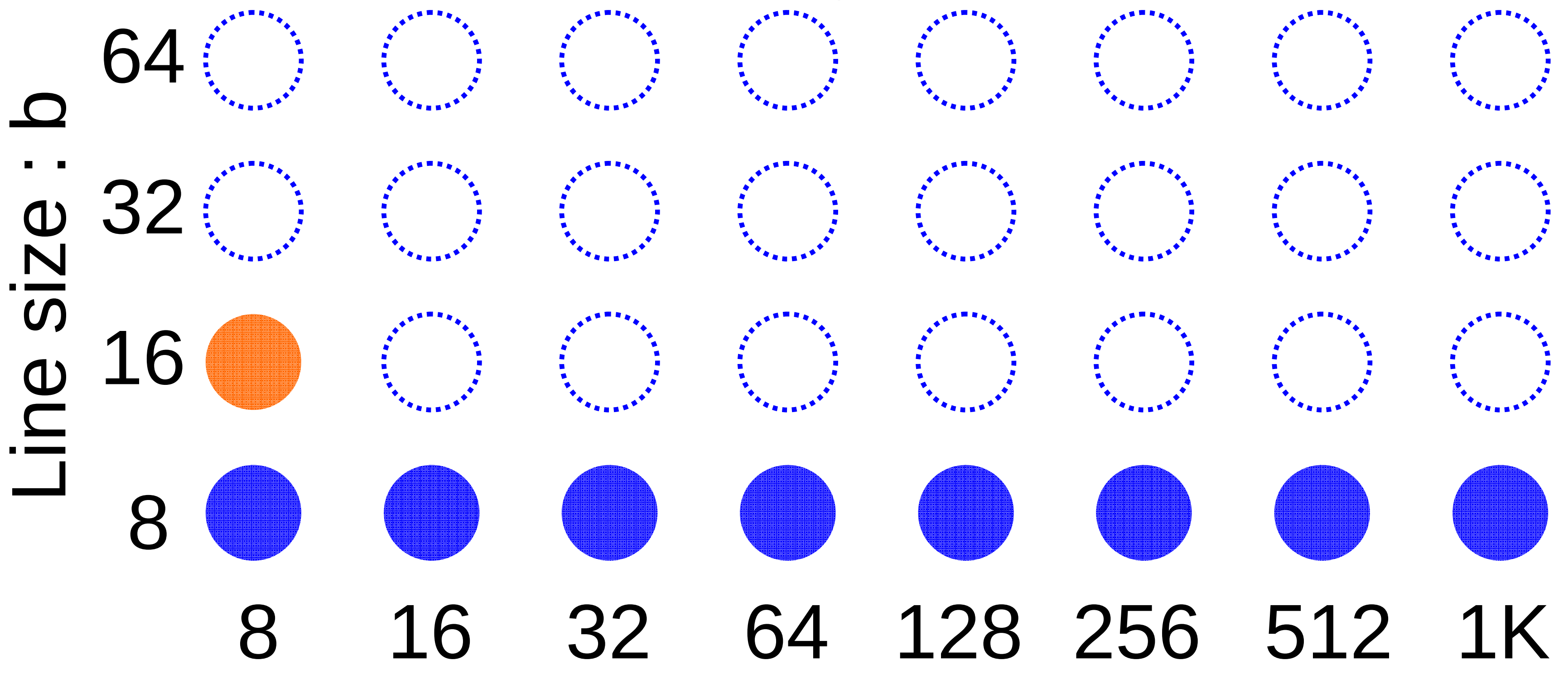
The associativity : the maximum



CRCB2

We respect to the memory access m_a

The associativity : the maximum



Cache set size : s

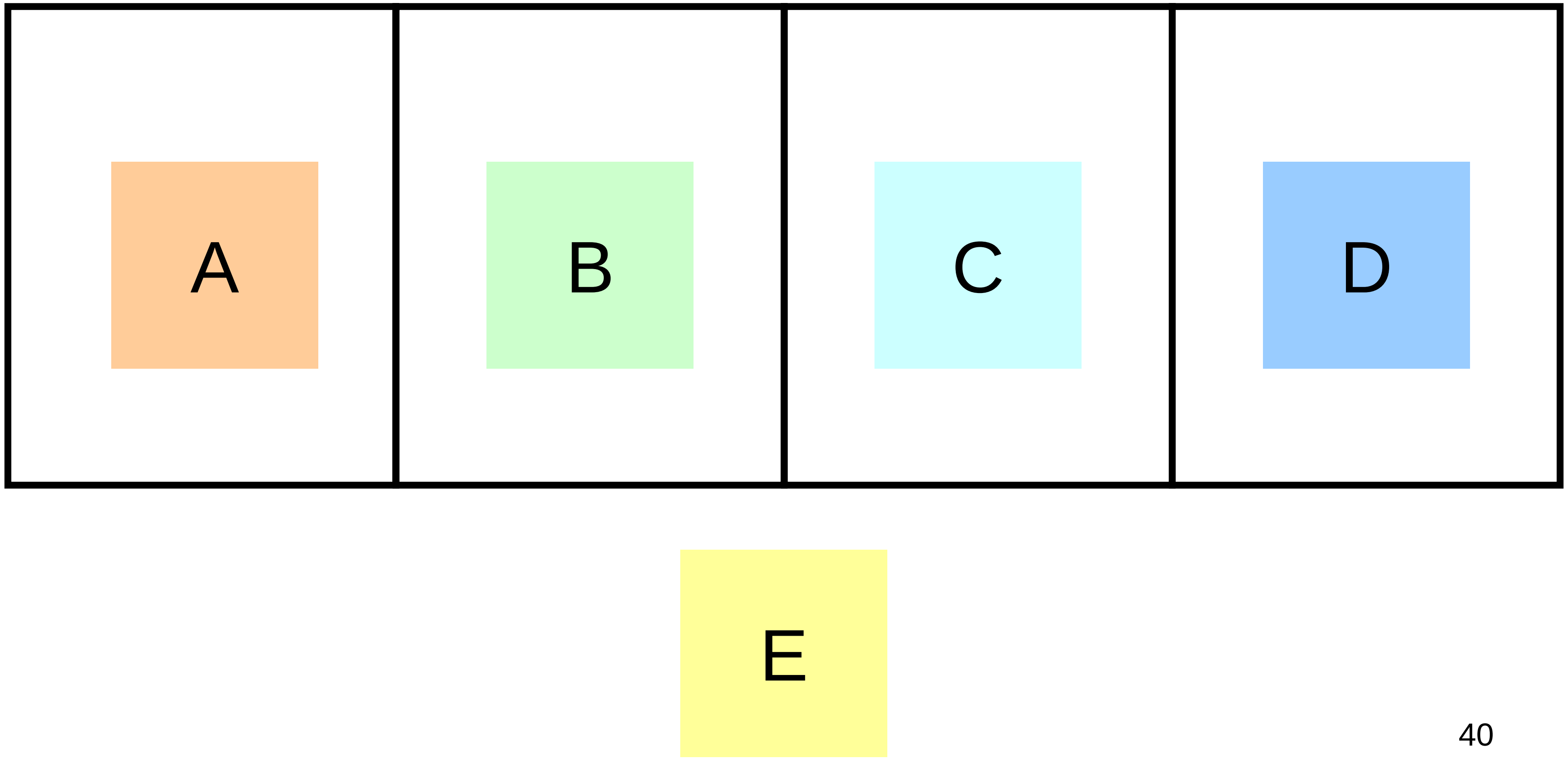
Cache data replacement

- Janapsatya’s approach[6]
  - Assume LRU replacement
  - Must LRU replacement
- CRCB1 and CRCB2
  - Do not assume any cache replacement
  - Can use any cache replacement

[6]：A.Janapsatya et al. , in Proc. ASP-DAC, 2006.

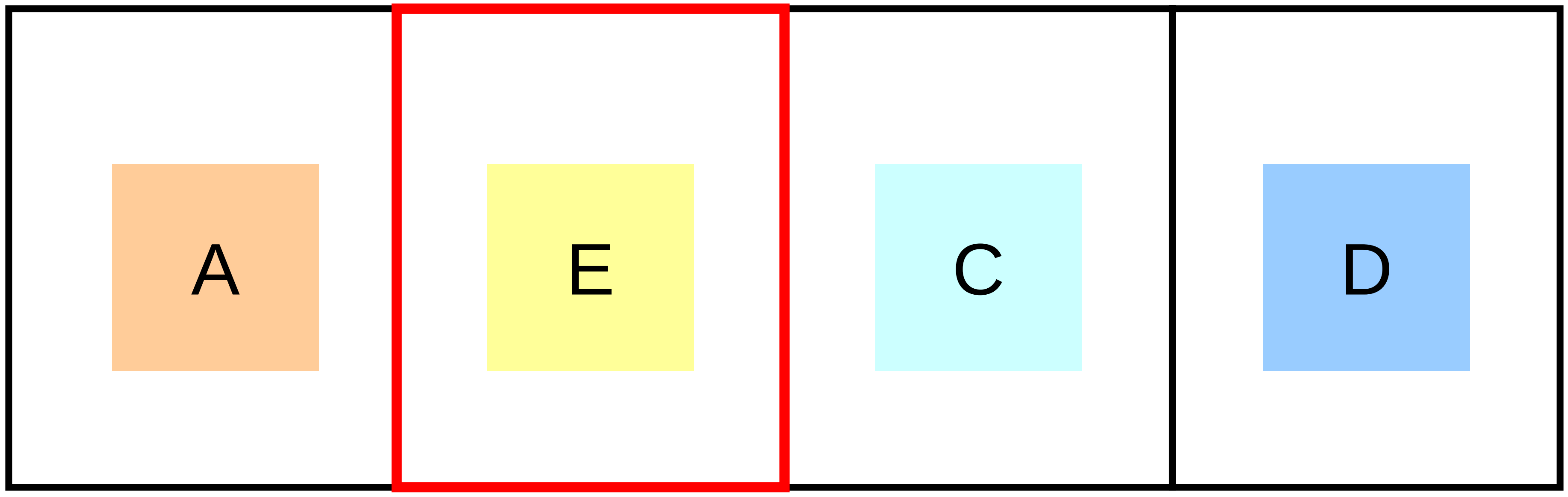
Cache data replacement

Associativity : a = 4

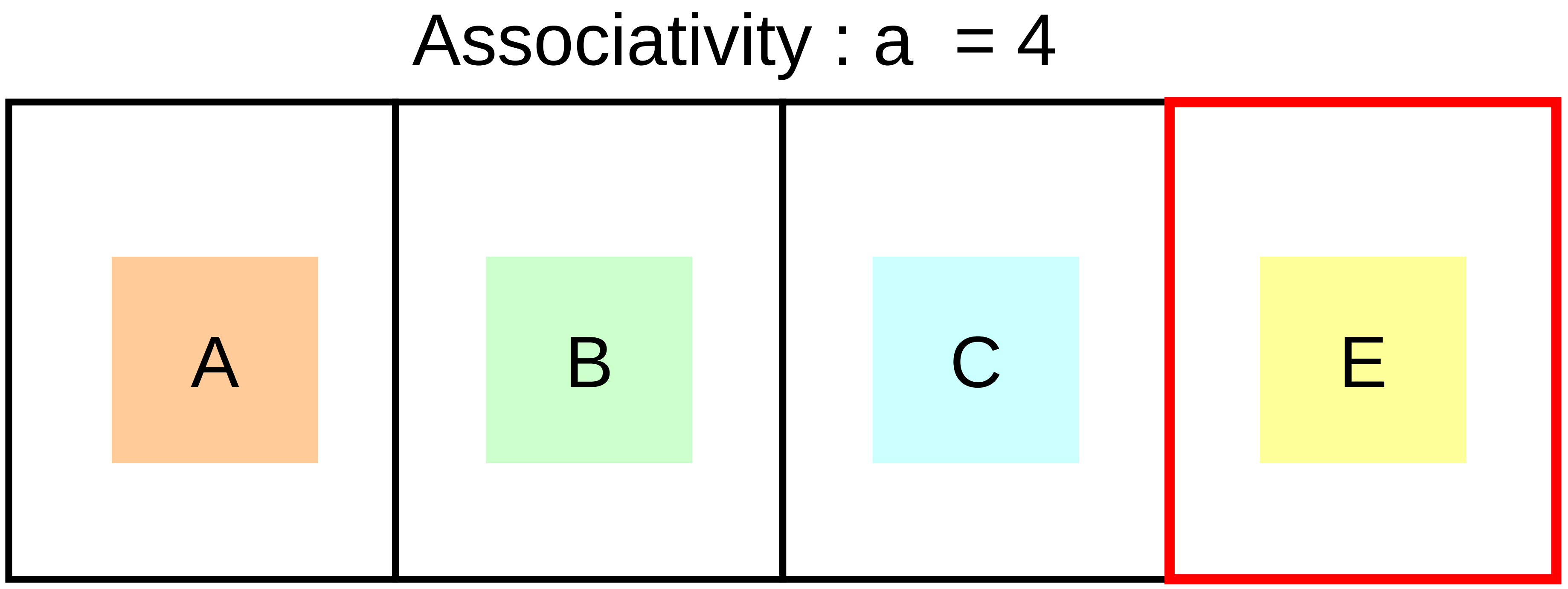


Cache data replacement

Associativity : a = 4

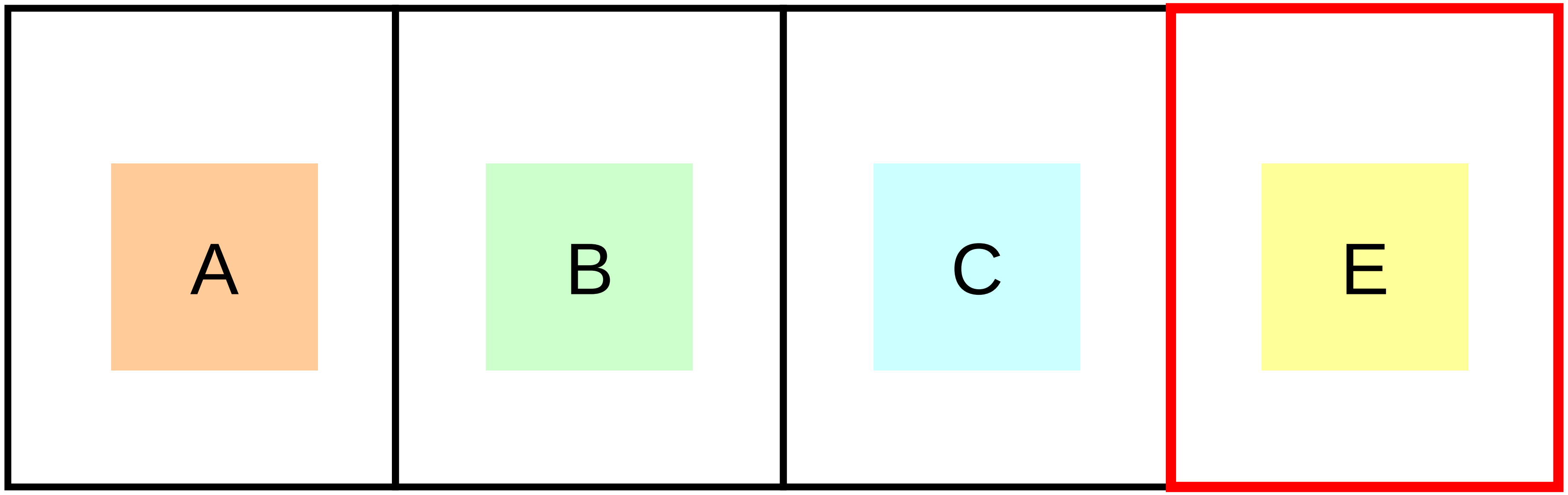


Cache data replacement



Cache data replacement

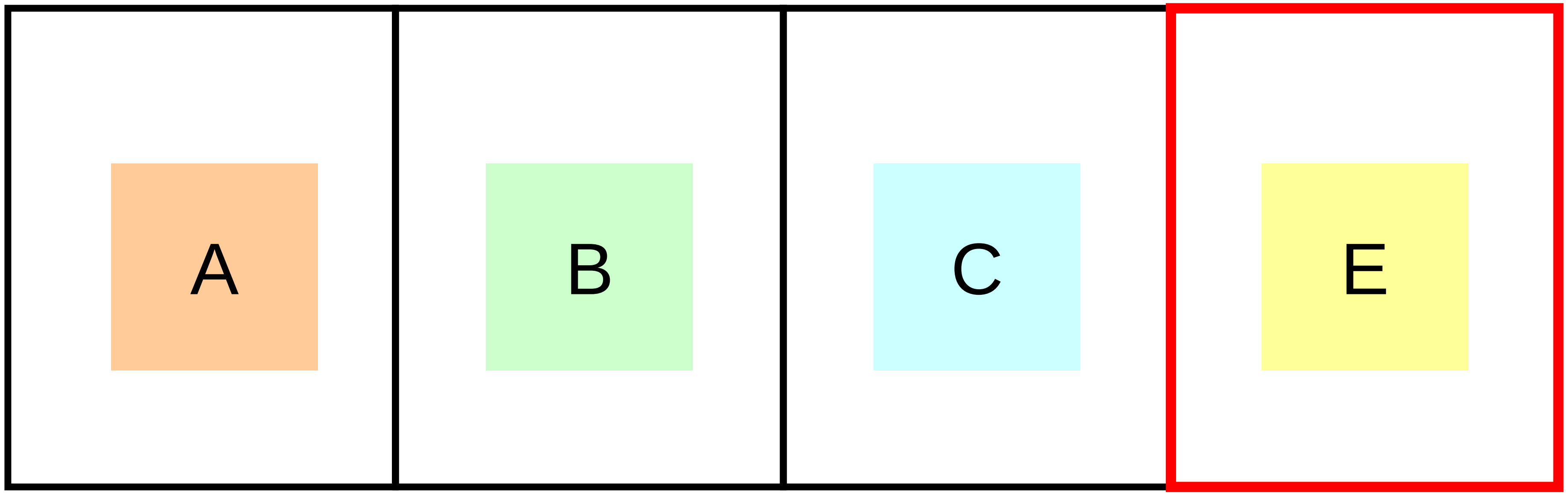
Associativity : a = 4



In any cache data replacement,
the last accessed data in each set is saved

Cache data replacement

Associativity : a = 4



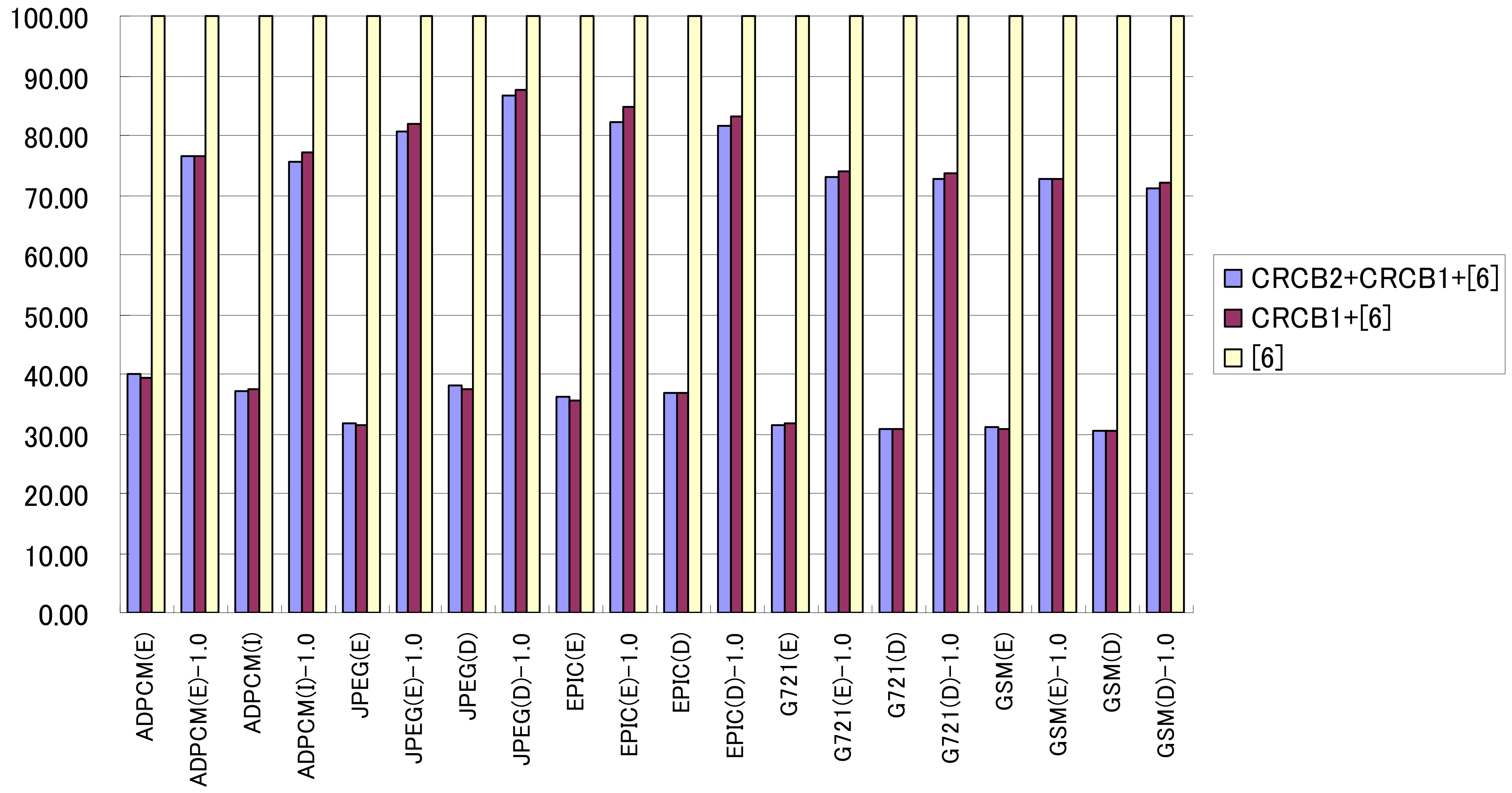
CRCB1 and CRCB2
We can use any cache data replacement

Our system

- Trace files are generated by using SimpleScalar/PISA3.0d[1]
- Use the cache models from CACTI 4.2[10]
- Run on Intel Xeon processor at 3.40[GHz] at 4[GB] memory
- Use several application programs from the MediaBench Suite[7]

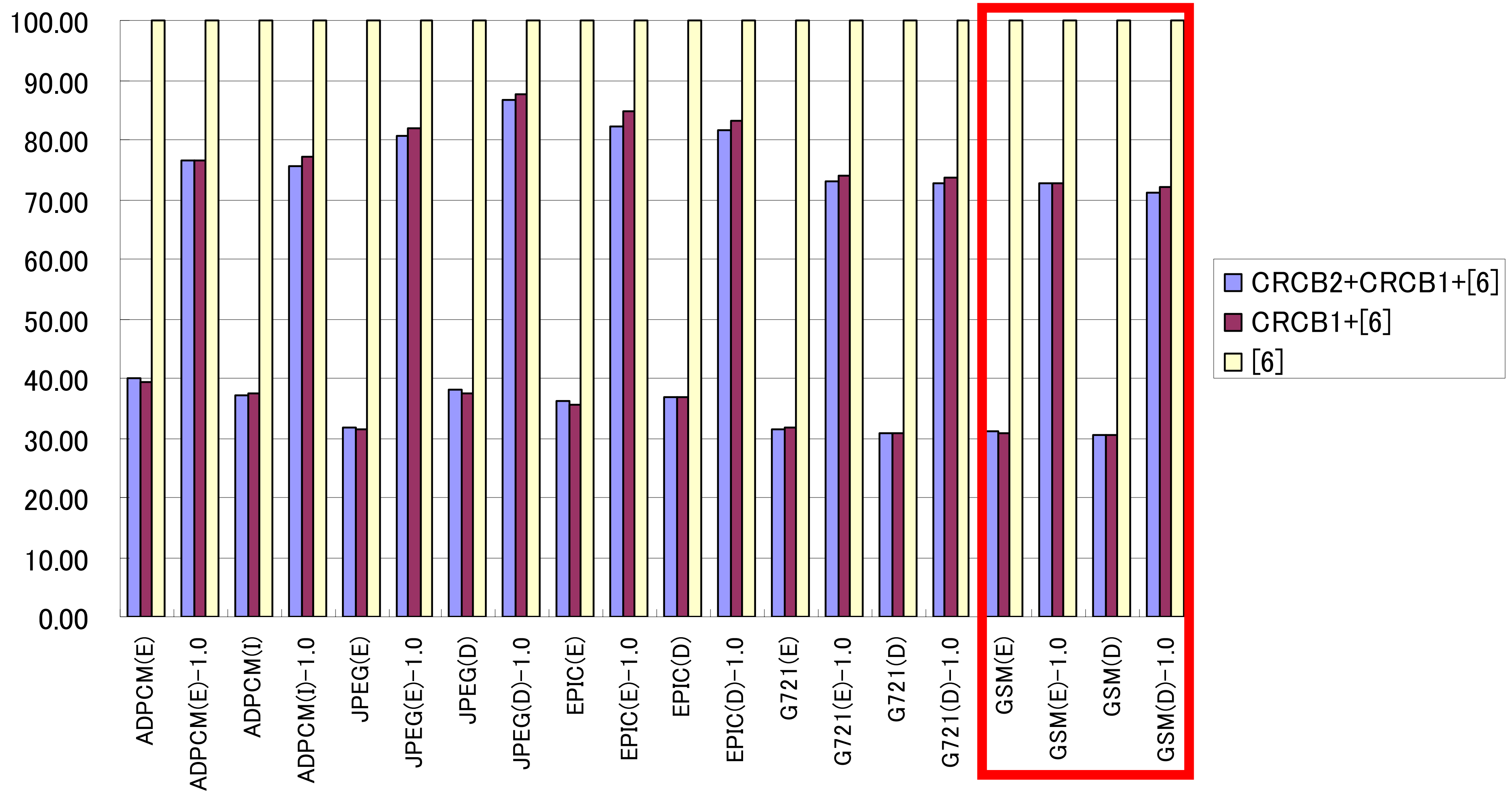
[1]：A. Austin et al. , IEEE Trans. Computer, 2002.
[7]：C. Lee et al. , in Proc. Annual International Symposium on Microarchitecture, 1997.
[10]：D. Tarjan et al. , HP Lab Technical Reports, 2006.

Experimental Results



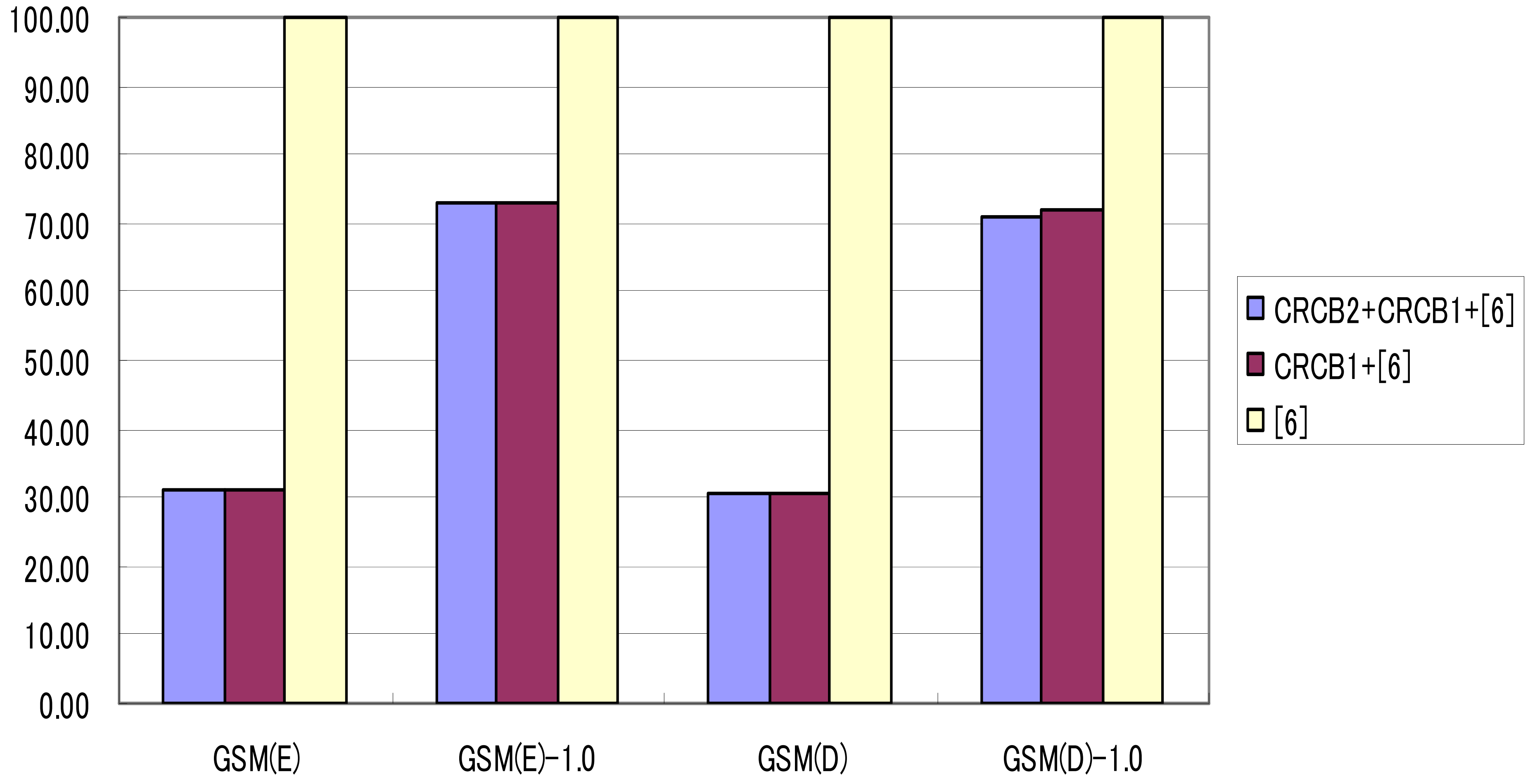
[6]：A.Janapsatya et al. , in Proc. ASP-DAC, 2006.

Experimental Results



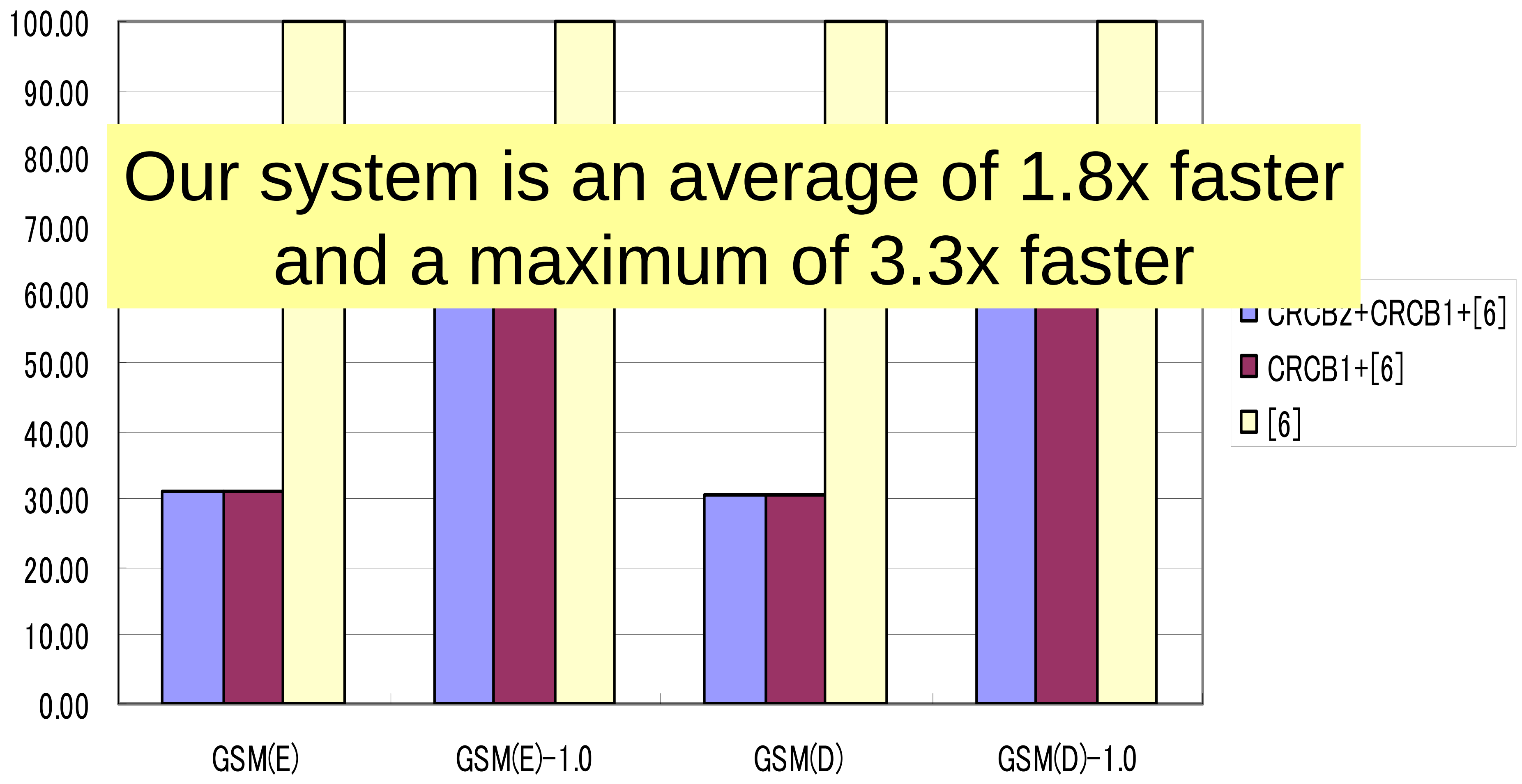
[6]：A.Janapsatya et al. , in Proc. ASP-DAC, 2006.

Experimental Results



[6]：A.Janapsatya et al. , in Proc. ASP-DAC, 2006.

Experimental Results



Our system is an average of 1.8x faster and a maximum of 3.3x faster

[6]：A.Janapsatya et al. , in Proc. ASP-DAC, 2006.

Conclusion

Exact and fast L1 cache simulation

Skipped cache simulations

CRCB1 and CRCB2

Our system is an average of 1.8x faster

Future work

Extending our approaches

to explore L2 cache and scratch pad memory

Thank you