2025年度(令和7年)版

Ver. 2025-10-14a

Course number: CSC.T363

コンピュータアーキテクチャ
Computer Architecture

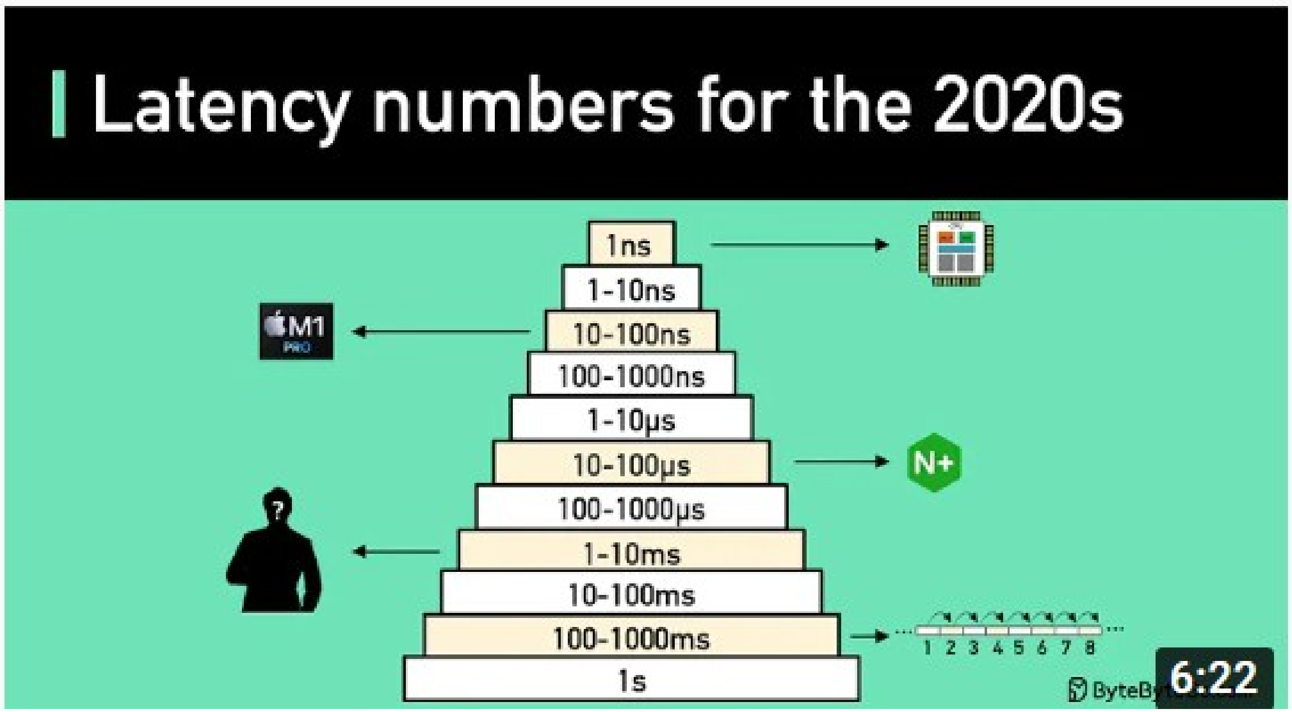
5. キャッシュ：セットアソシアティブ方式
Caches: Set-Associative

www.arch.cs.titech.ac.jp/lecture/CA/
Tue 13:30-15:10, 15:25-17:05
Fri 13:30-15:10

吉瀬 謙二 情報工学系
Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

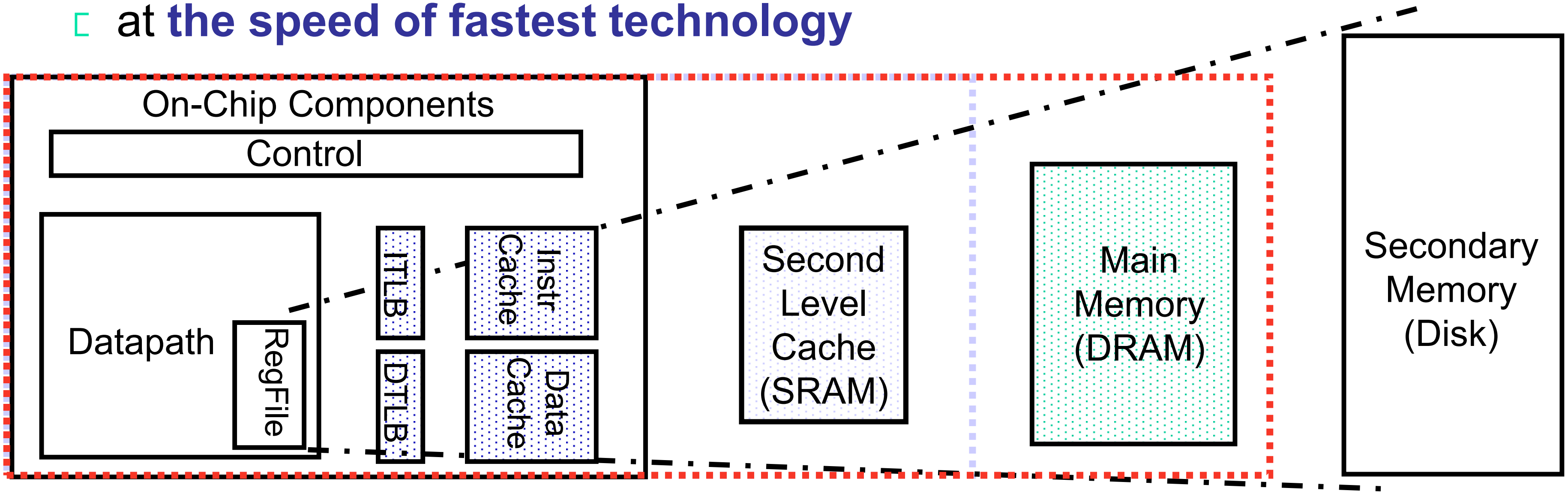
参考

- Latency Numbers Programmer Should Know
  - https://www.youtube.com/watch?v=FqR5vESuKe0



A Typical Memory Hierarchy

- By taking advantage of **the principle of locality** (局所性)
  - Present **much memory** in **the cheapest technology**
  - at **the speed of fastest technology**



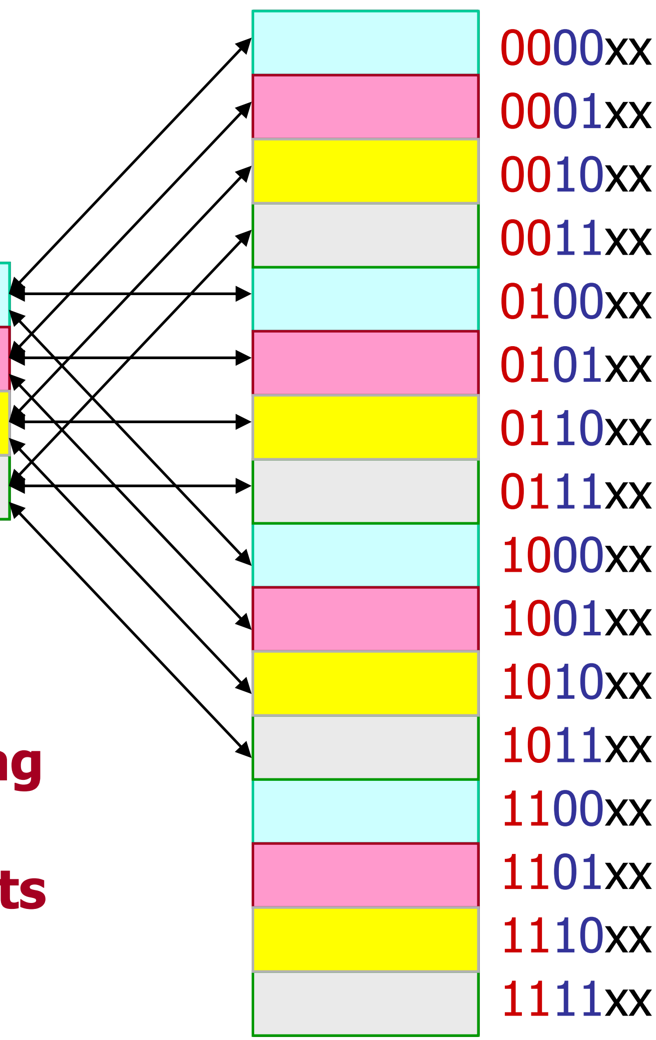
Speed (%cycles):	½'s	1's	10's	100's	1,000's
Size (bytes):	100's	K's	10K's	M's	G's to T's
Cost:	highest				lowest

TLB: Translation Lookaside Buffer

Caching: Direct mapped (First Example)

Cache

Index	Valid	Tag	Data
00			
01			
10			
11			



Main Memory

Two low order bits define the byte in the word (32-b words)

Q1: Is it there?

Compare the cache **tag** to the **high order 2 memory address bits** to tell if the memory block is in the cache

Q2: How do we find it?

Use **next 2 low order memory address bits** – the **index** – to determine which cache block

(block address) modulo (# of blocks in the cache)

Direct Mapped Cache Example

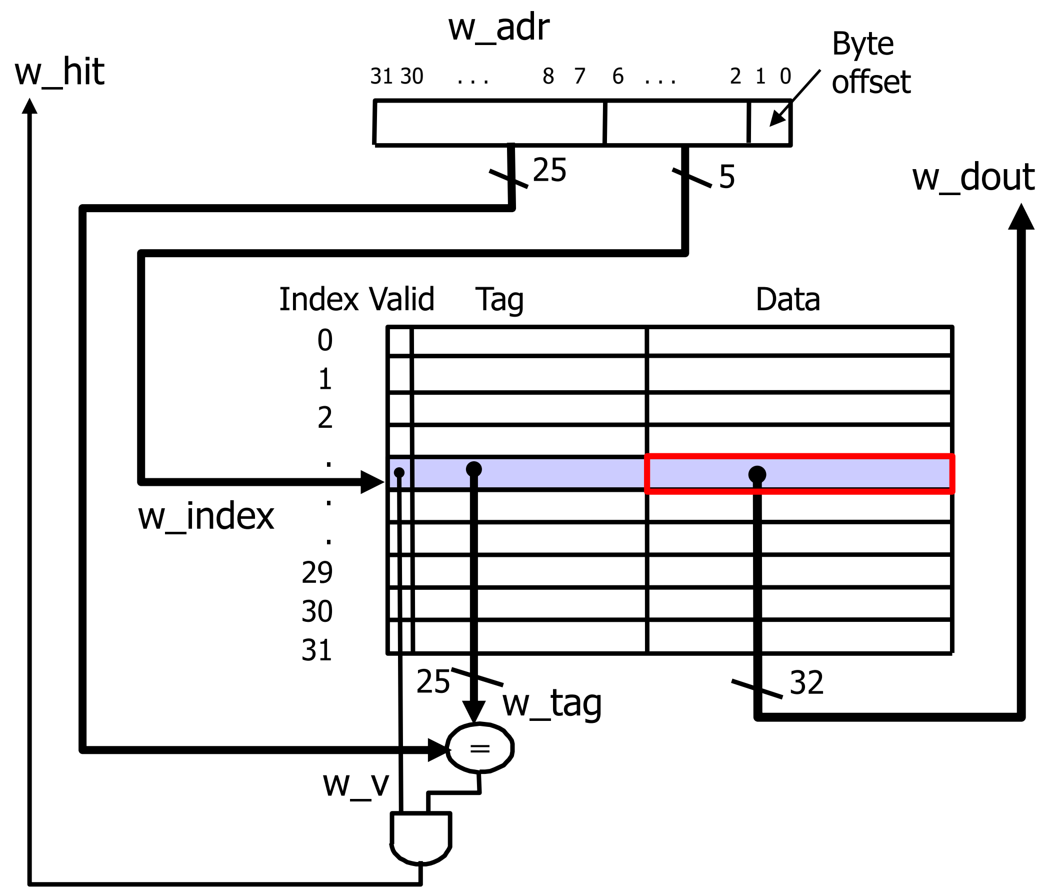
- One word/block, cache size = 1K words

```
module m_cache_direct_mapped_32 (
  input  wire        w_clk,
  input  wire        w_we,
  input  wire [31:0] w_adr,
  input  wire [4:0]  w_wadr,
  input  wire [57:0] w_wd,
  output wire        w_hit,
  output wire [31:0] w_dout
);

  reg [57:0] mem [0:31];
  integer i; initial for (i=0; i<32; i=i+1) mem[i] = 0;

  wire [4:0]  w_index = w_adr[6:2];
  wire        w_v;
  wire [24:0] w_tag;
  assign {w_v, w_tag, w_dout} = mem[w_index];
  assign w_hit = w_v & (w_adr[31:7]==w_tag);

  always @(posedge w_clk) if (w_we) mem[w_wadr] <= w_wd;
endmodule
```



Direct Mapped Cache Example

- One word/block, cache size = 1K words

```
module m_cache_direct_mapped_32 (
  input  wire        w_clk,
  input  wire        w_we,
  input  wire [31:0] w_adr,
  input  wire [4:0]  w_wadr,
  input  wire [57:0] w_wd,
  output wire        w_hit,
  output wire [31:0] w_dout
);

  reg [57:0] mem [0:31];
  integer i; initial for (i=0; i<32; i=i+1) mem[i] = 0;

  wire [4:0]  w_index = w_adr[6:2];
  wire        w_v;
  wire [24:0] w_tag;
  assign {w_v, w_tag, w_dout} = mem[w_index];
  assign w_hit = w_v & (w_adr[31:7]==w_tag);

  always @(posedge w_clk) if (w_we) mem[w_wadr] <= w_wd;
endmodule
```

1R/1W memory

```
module m_cache_direct_mapped_32_v2 (
  input  wire        w_clk,
  input  wire        w_we,
  input  wire [31:0] w_adr,
  input  wire [57:0] w_wd,
  output wire        w_hit,
  output wire [31:0] w_dout
);

  reg [57:0] mem [0:31];
  integer i; initial for (i=0; i<32; i=i+1) mem[i] = 0;

  wire [4:0]  w_index = w_adr[6:2];
  wire        w_v;
  wire [24:0] w_tag;
  assign {w_v, w_tag, w_dout} = mem[w_index];
  assign w_hit = w_v & (w_adr[31:7]==w_tag);

  always @(posedge w_clk) if (w_we) mem[w_index] <= w_wd;
endmodule
```

1RW memory

Sources of Cache Misses

- Compulsory (初期参照ミス, cold start or process migration, first reference):
  - First access to a block, “cold” fact of life, not a whole lot you can do about it
  - If you are going to run “millions” of instruction, compulsory misses are insignificant
- Conflict (競合性ミス, collision):
  - Multiple memory locations mapped to the same cache location
  - Solution 1: increase cache size
  - Solution 2: increase **associativity**
- Capacity (容量性ミス):
  - Cache cannot contain all blocks accessed by the program
  - Solution: increase cache size

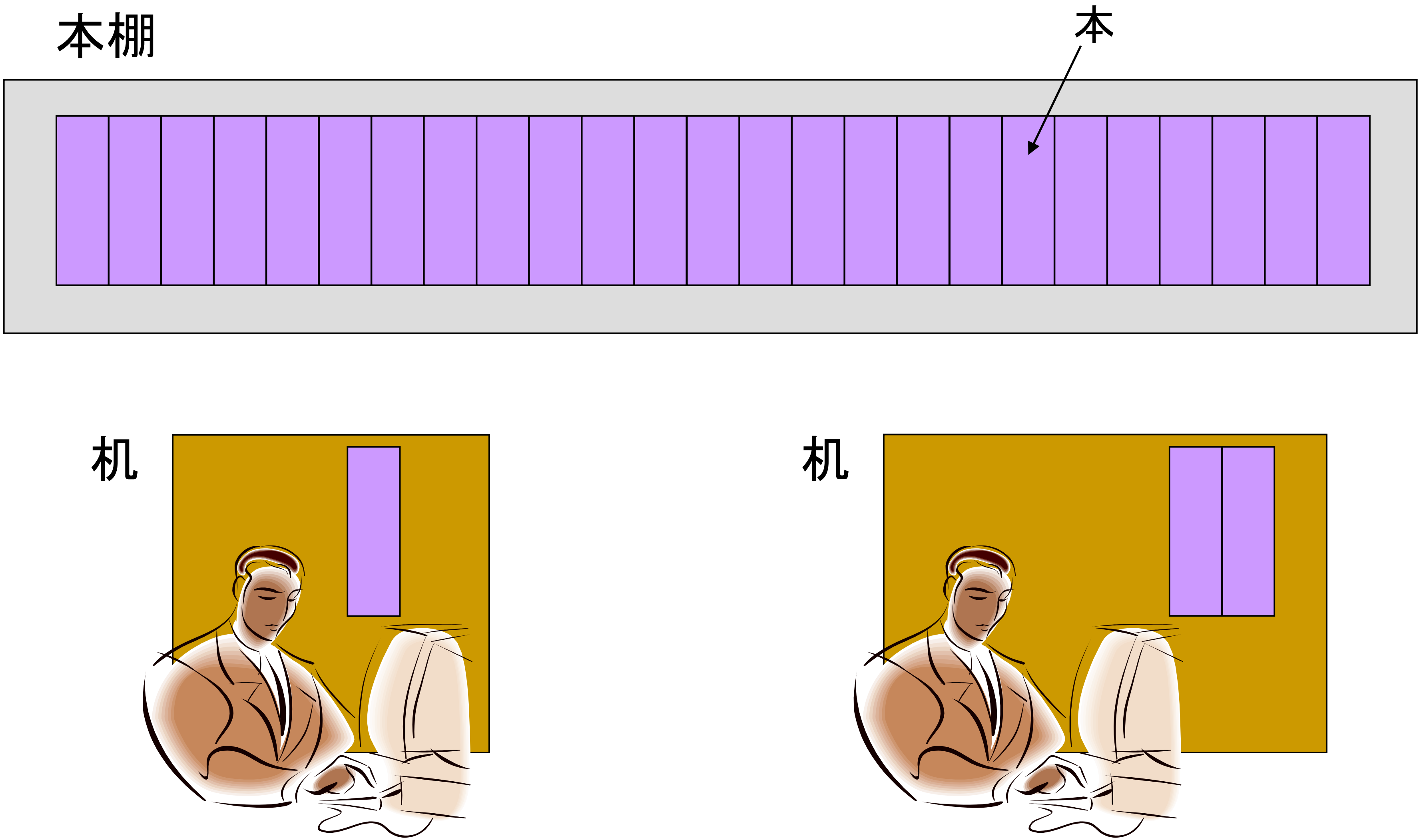
Reducing Cache Miss Rates, Associativity

Allow more flexible block placement

- In a **direct mapped cache** a memory block maps to exactly **one cache block**
- At the other extreme, could allow a memory block to be mapped to any cache block – **fully associative cache**

- A compromise is to divide the cache into **sets** each of which consists of **n** "**ways**" (**n-way set associative**).
  A memory block maps to a unique set and can be placed in any way of that set (so there are **n choices**)

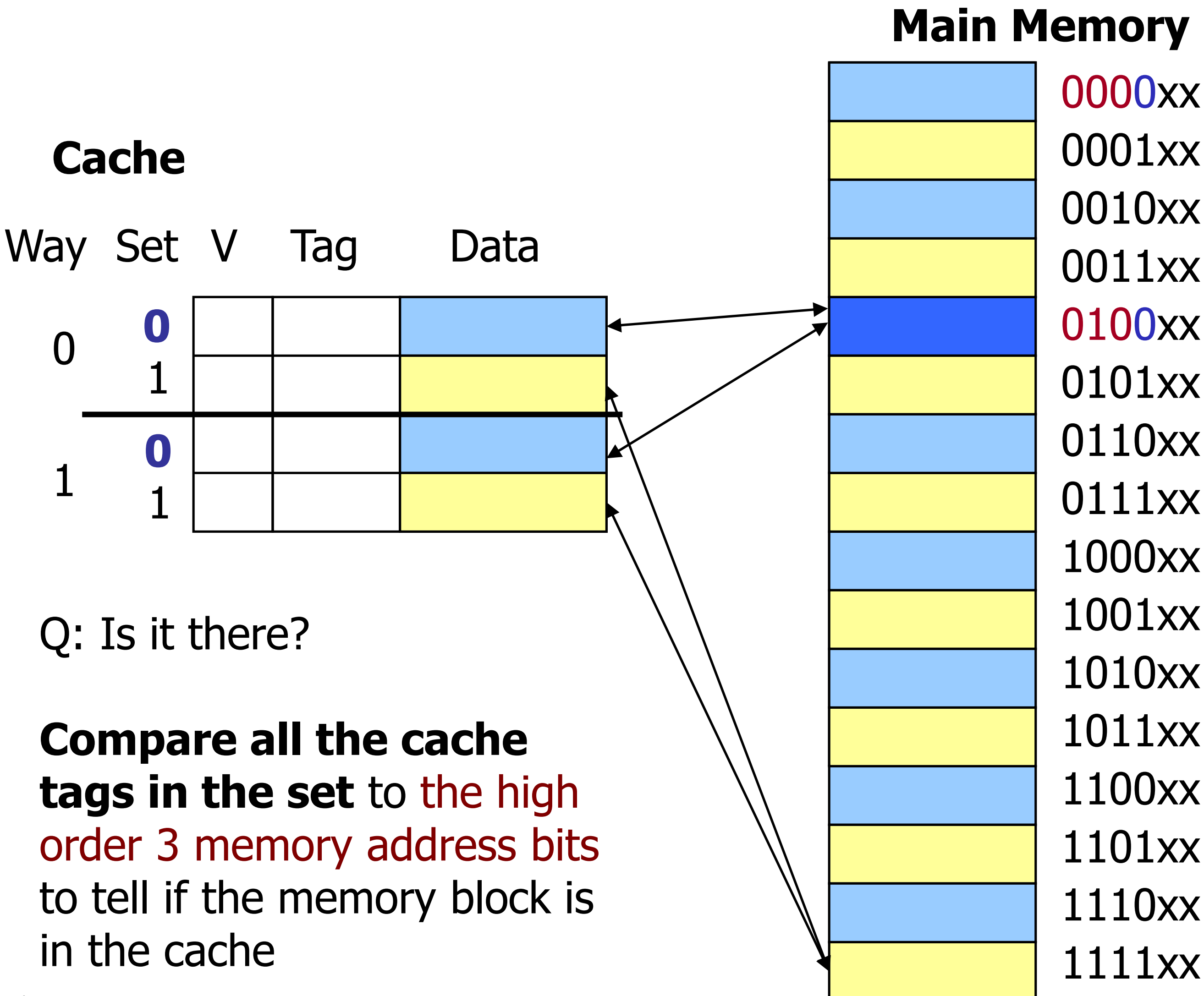
Cache Associativity



Direct Mapped

Set Associative

Set Associative Cache Example



Two low order bits define the byte in the word (32-b words)
One word blocks

Q: How do we find it?

Use next 1 low order memory address bit to determine which cache set

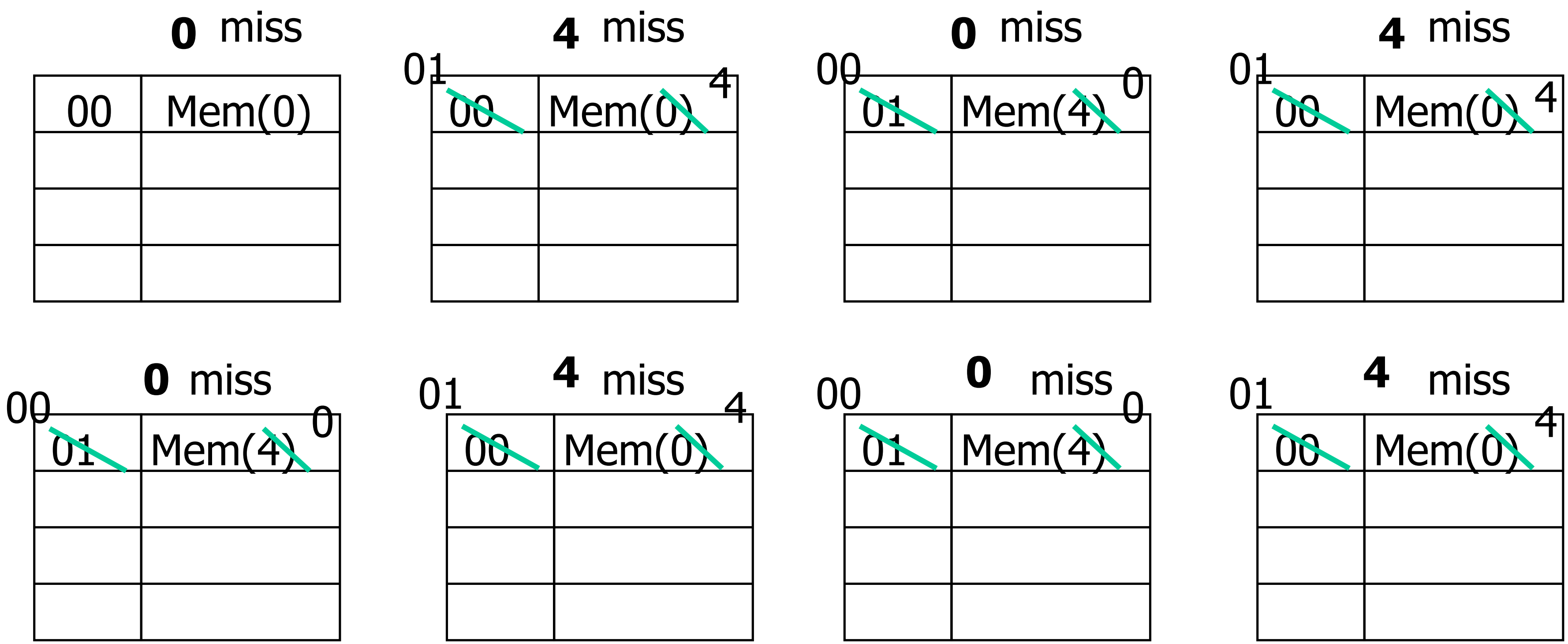
Q: Is it there?

Compare all the cache tags in the set to the high order 3 memory address bits to tell if the memory block is in the cache

Another Reference String Mapping (Direct Mapped)

- Consider the main memory word reference string

0 4 0 4 0 4 0 4



- 8 requests, 8 misses
  - Ping pong effect due to **conflict** misses - two memory locations that map into the same cache block

Another Reference String Mapping (Set Associative)

Consider the main memory word reference string

0 4 0 4 0 4 0 4

Start with an empty cache –
all blocks initially marked as not valid

0 miss

000	Mem(0)

4 miss

000	Mem(0)
010	Mem(4)

0 hit

000	Mem(0)
010	Mem(4)

4 hit

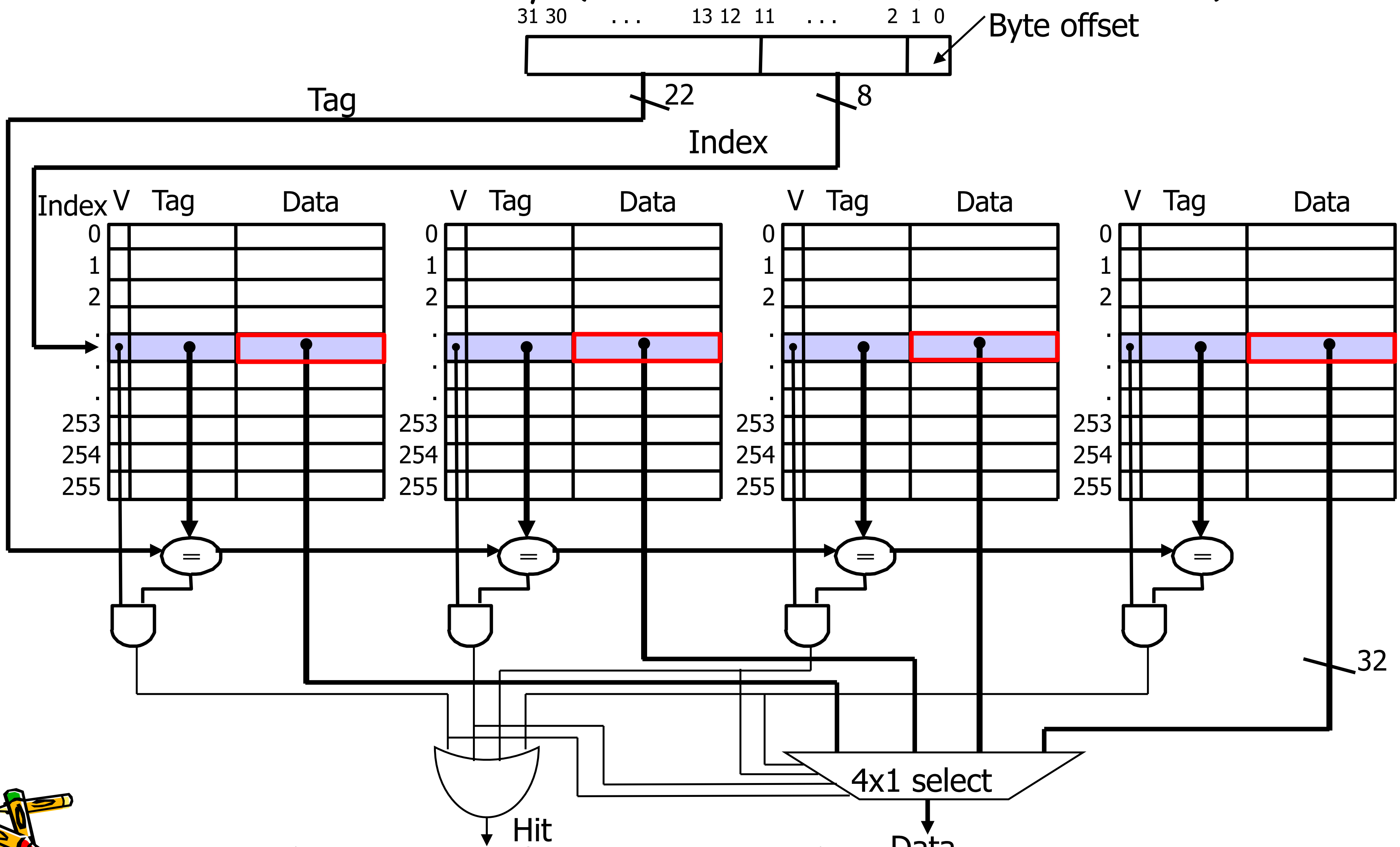
000	Mem(0)
010	Mem(4)

- 8 requests, 2 misses

- Solves the **ping pong effect** in a direct mapped cache due to conflict misses

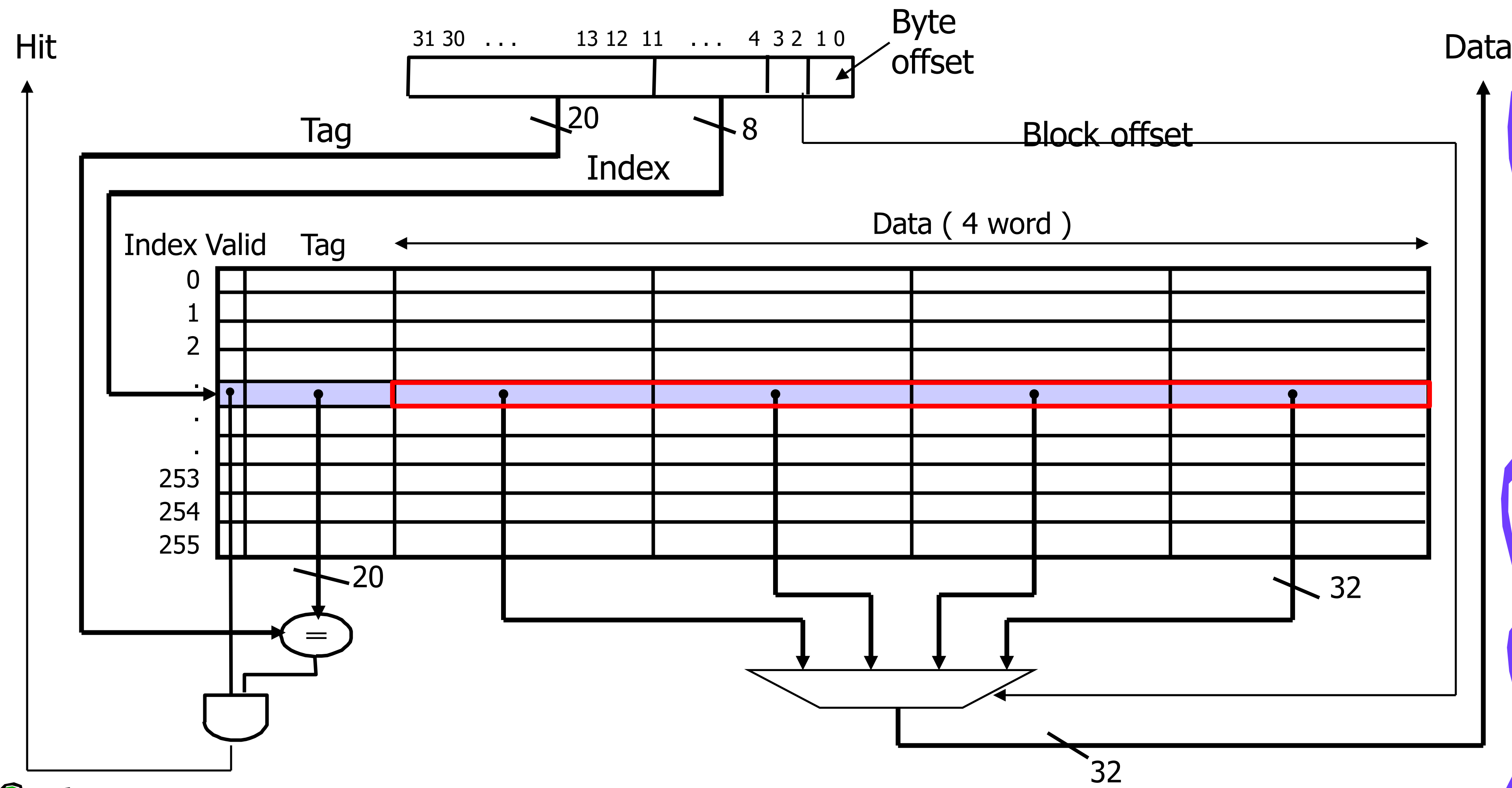
Four-Way Set Associative Cache

2^8 = 256 sets each with four ways (each with one block, one word / block)



Multiword Block Direct Mapped Cache

- Four words/block, cache size = 1K words



Range of Set Associative Caches

For a fixed size cache

Used for tag compare

Selects the set

Selects the word in the block

Tag	Index	Block offset	Byte offset

Increasing associativity

Decreasing associativity

Fully associative
(only **one set**)
Tag is all the bits except block and byte offset

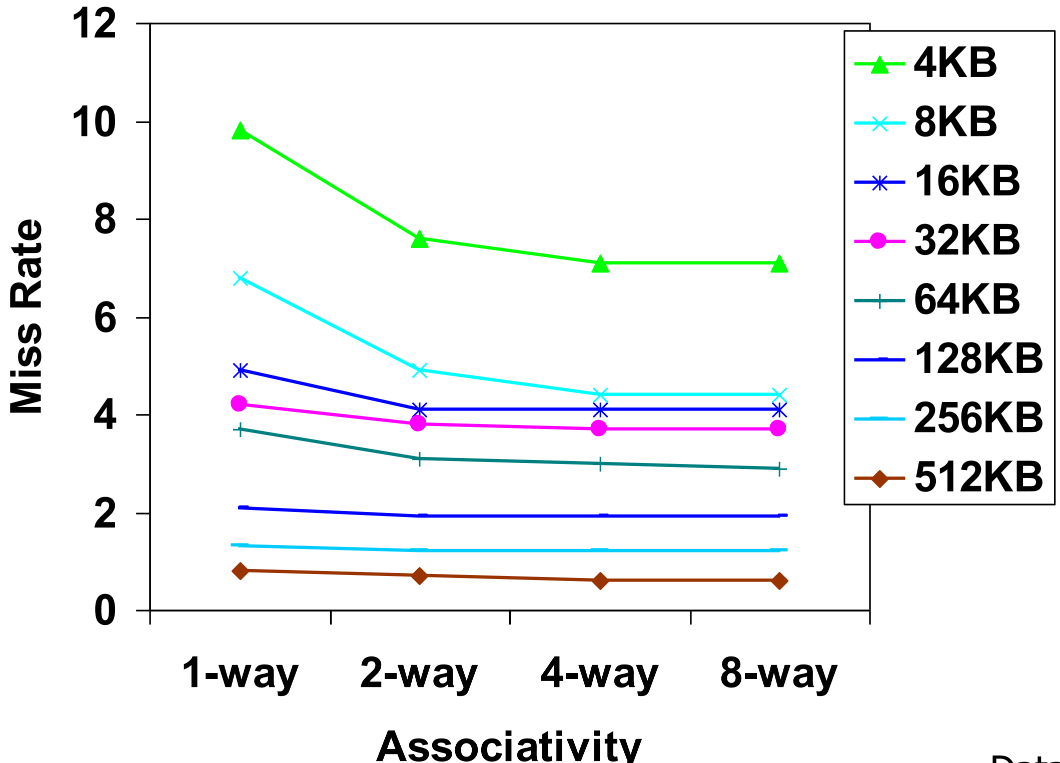
Direct mapped
(only **one way**)
Smaller tags

Costs of Set Associative Caches

- **N-way set associative** cache costs
  - N comparators (delay and area)
  - MUX delay (set selection) before data is available
  - Data available **after** set selection and Hit/Miss decision.
- **When a miss occurs**,
  which way's block do we pick **for replacement** ?
  - **Least Recently Used (LRU):**
    - the block replaced is the one that has been unused for the longest time
      - Must have hardware to keep track of when each way's block was used
      - For **2-way set associative**, takes **one bit per set** →
        set the bit when a block is referenced
        (and reset the other way's bit)
  - **Random**

Benefits of Set Associative Caches

The choice of direct mapped or set associative depends on **the cost of a miss** versus **the cost of implementation**



Data from Hennessy & Patterson, *Computer Architecture*, 2003

- Largest gains are in going from **direct mapped** to **2-way**

Clock rate is mainly determined by

- Switching speed of gates (transistors)
- The number of levels of gates
    - The maximum number of gates cascaded in series in any combinational logics.
    - In this example, the number of levels of gates is 3.
- Wiring delay and fanout
- The slowest of all paths is called the critical path