CACHE MEMORY

Program locality

- Temporal locality (Data)
  - Recently referenced information
- Spatial locality
  - Same address space will be referenced
- Sequential locality (instructions)
  - (spatial locality) next address will be used

Caches

- Unified cache
- Split cache
  - Instruction cache
  - Data cache

Tradeoffs

- Usually cache (L1 and L2) is within CPU chip.
- Thus, area is a concern
- Design alternatives:
  - Large I-cache
  - Equal area: I- and D-cache
  - Large unified cache
  - Multi-level split

Some terms

- Read: Any read access (by the processor) to cache –instruction (inst. fetch) or data
- Write: Any write access (by the processor) into cache
- Fetch: read to main mem. to load a block
- Access: any reference to memory

Miss Ratio

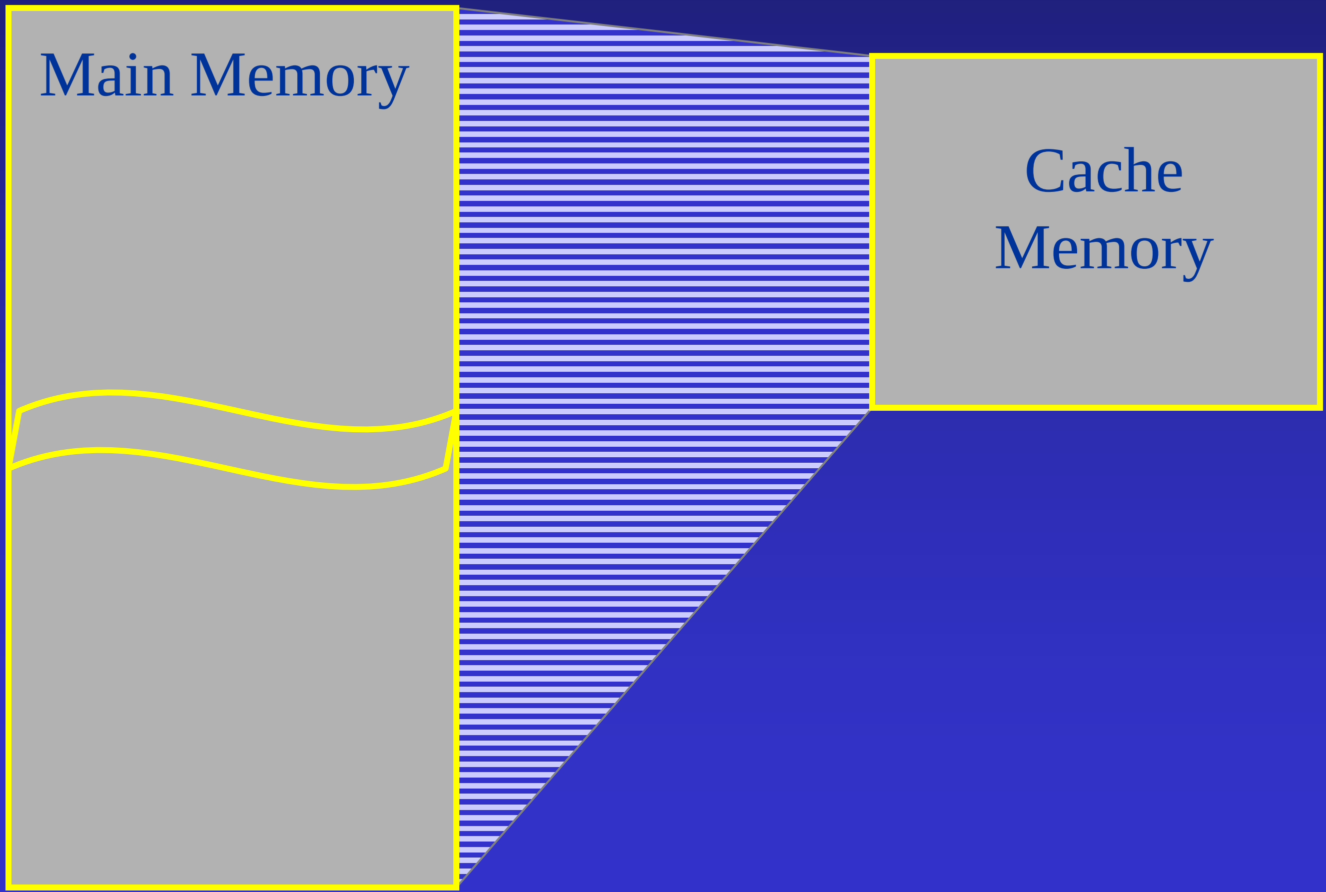
Number of references to cache not found in cache/total references to cache

$$MissRatio = \frac{Any\_ref_{miss}}{Any\_ref_{total}}$$

Number of I-cache references not found in cache/ instructions executed

$$MissRatio_{I-Cache} = \frac{Inst\_ref_{miss}}{Inst_{number}}$$

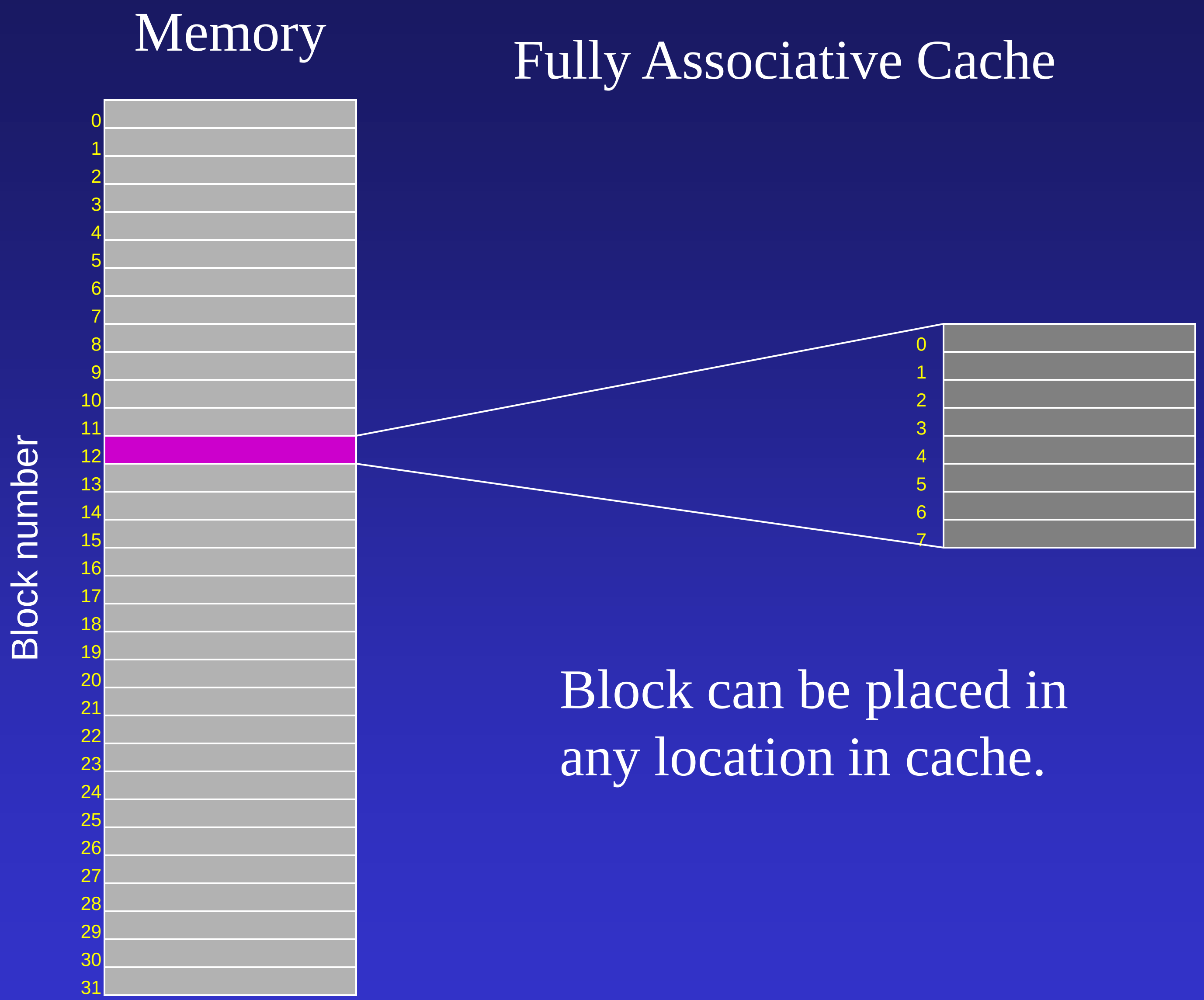
Placement Problem



Placement Policies

- WHERE to put a block in cache
- Mapping between main and cache memories.
- Main memory has a much larger capacity than cache memory.

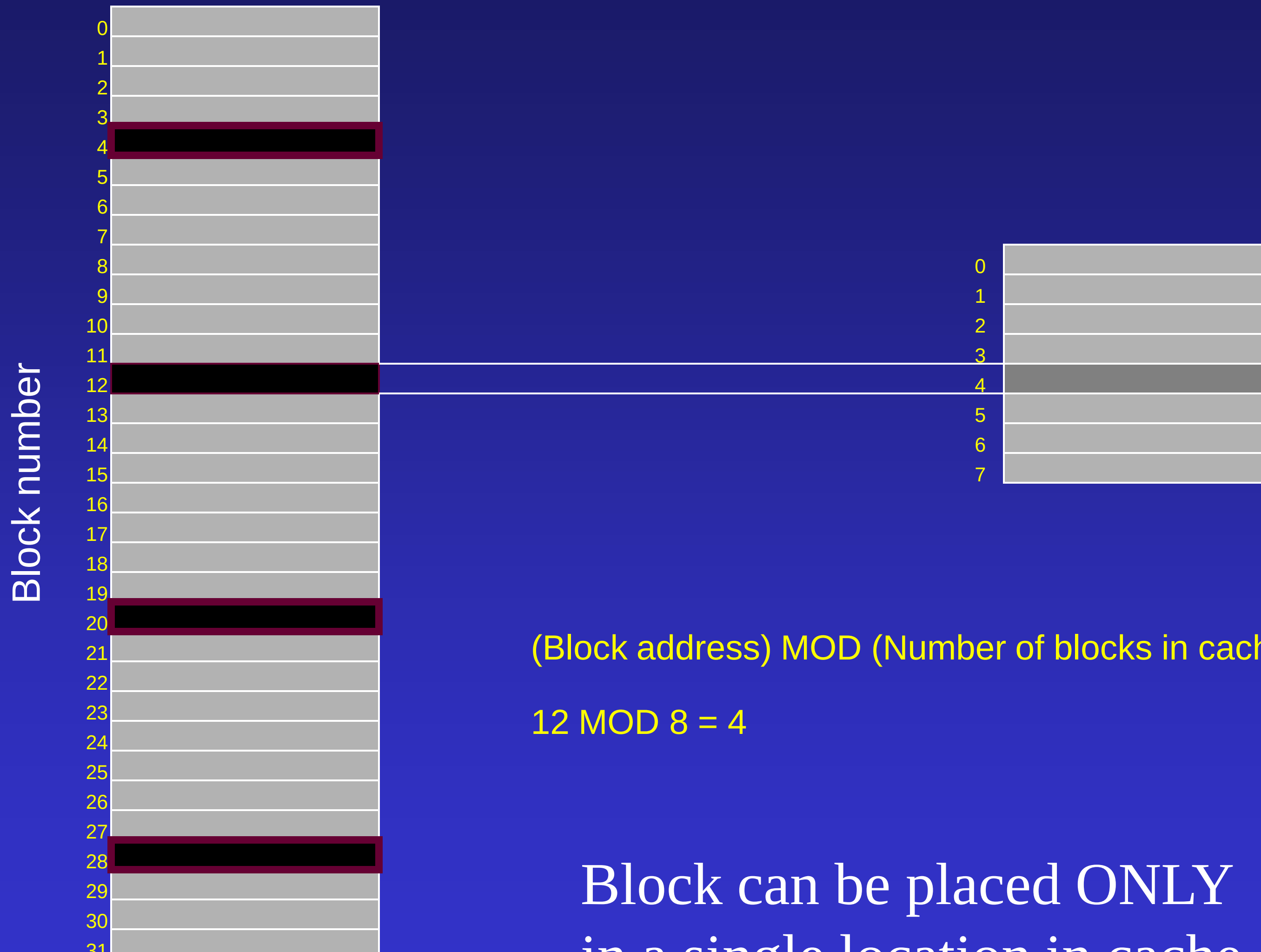
Fully Associative Cache



Block can be placed in any location in cache.

Memory

Direct Mapped Cache

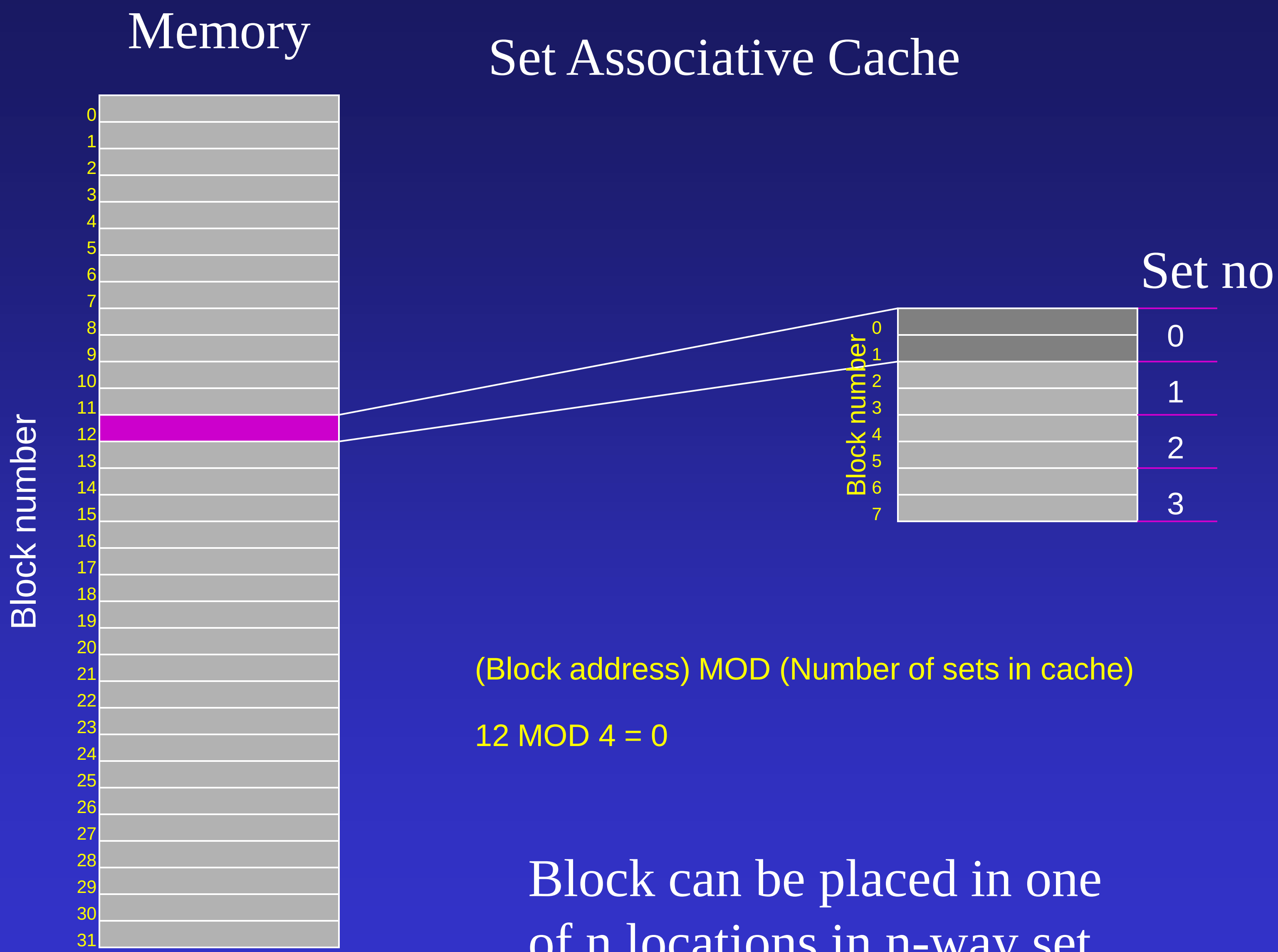


(Block address) MOD (Number of blocks in cache)

12 MOD 8 = 4

Block can be placed ONLY
in a single location in cache.

Set Associative Cache

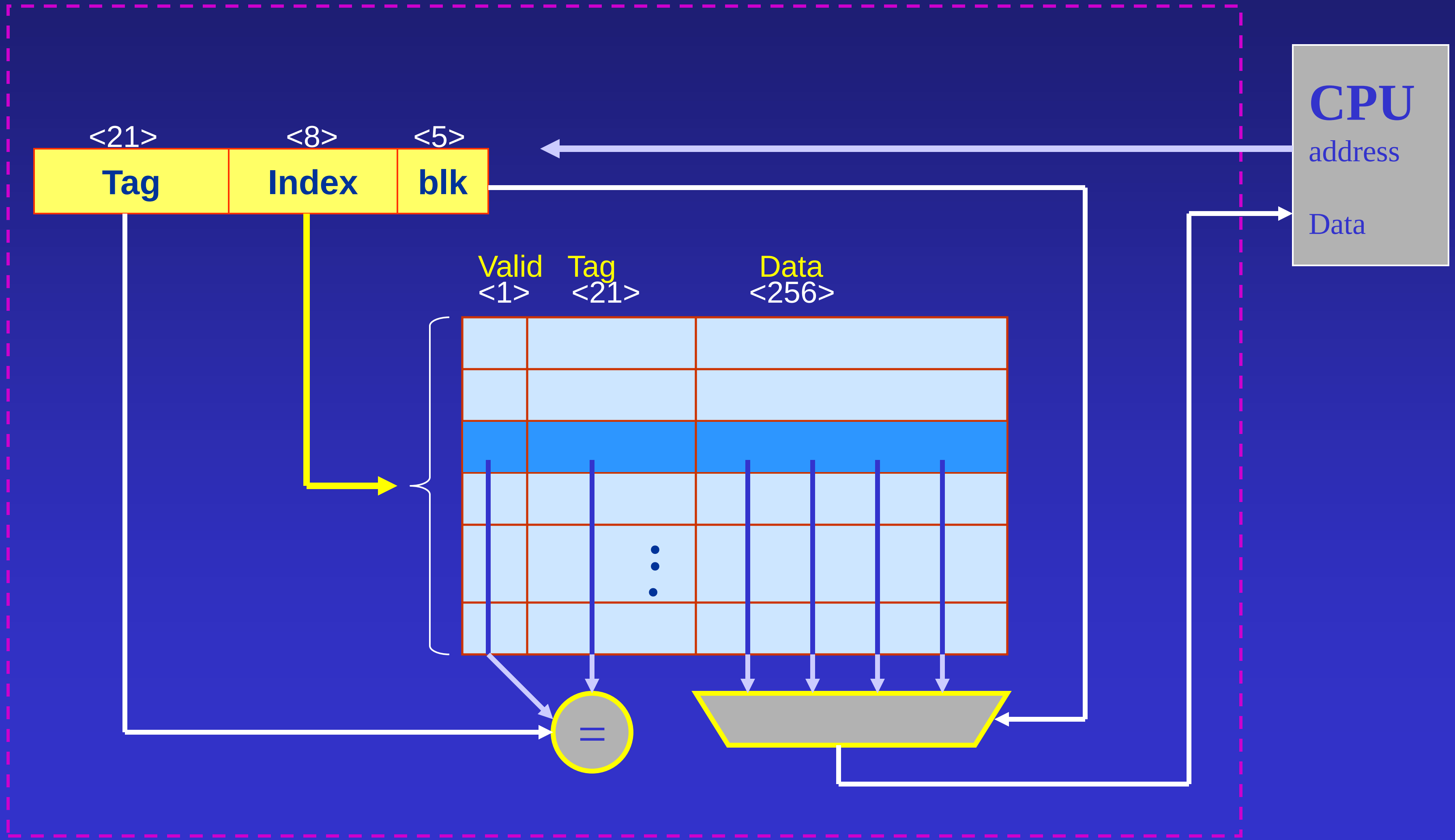


(Block address) MOD (Number of sets in cache)

12 MOD 4 = 0

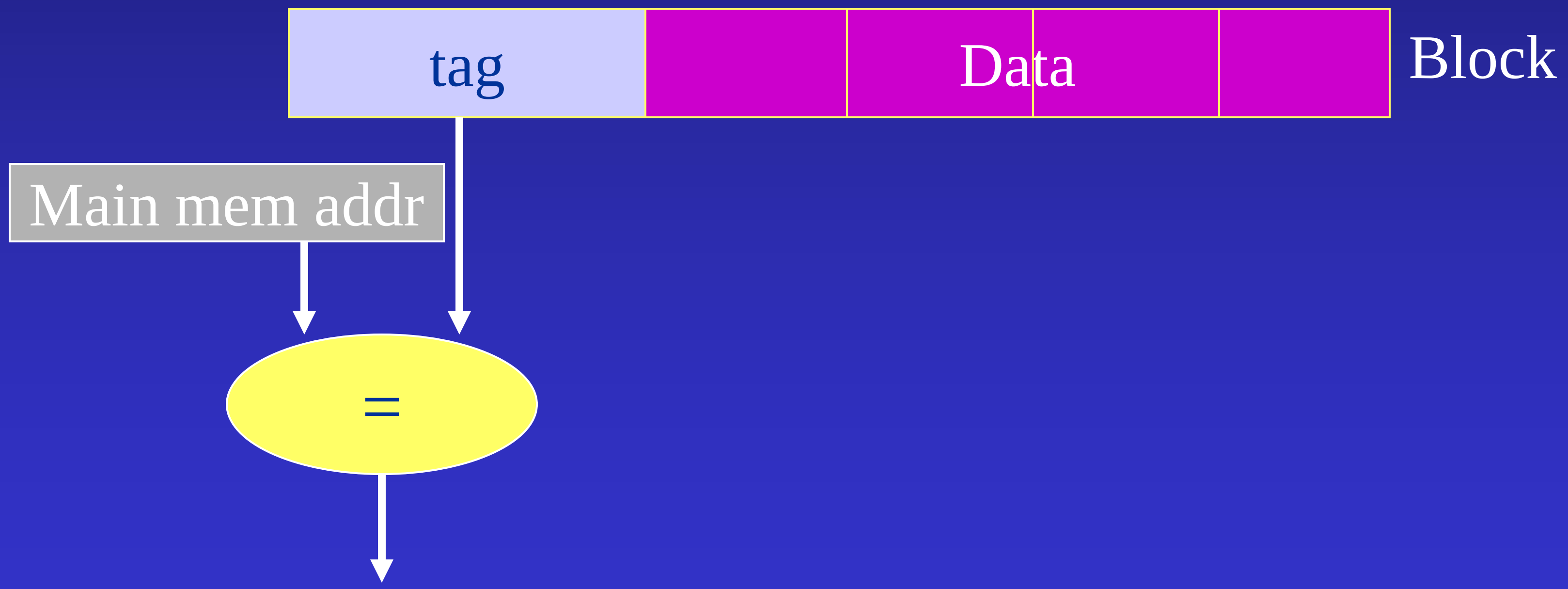
Block can be placed in one of n locations in n-way set associative cache.

Cache organization



TAG

- Contains the “sector name” of block



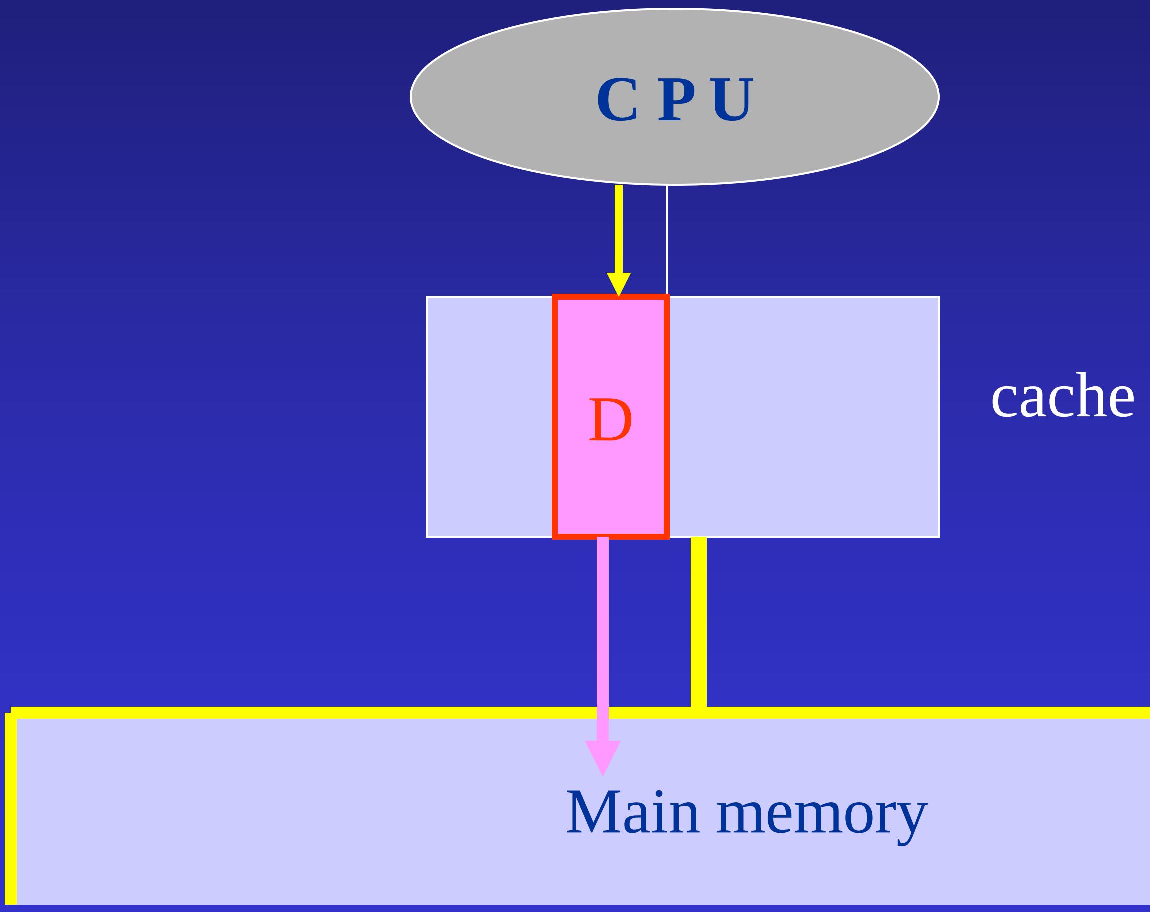
Valid bit(s)

- Indicates if block contains valid data
  - initial program load into main memory.
    - No valid data
  - Cache coherency in multiprocessors
    - Data in main memory could have been chanced by other processor (or process).

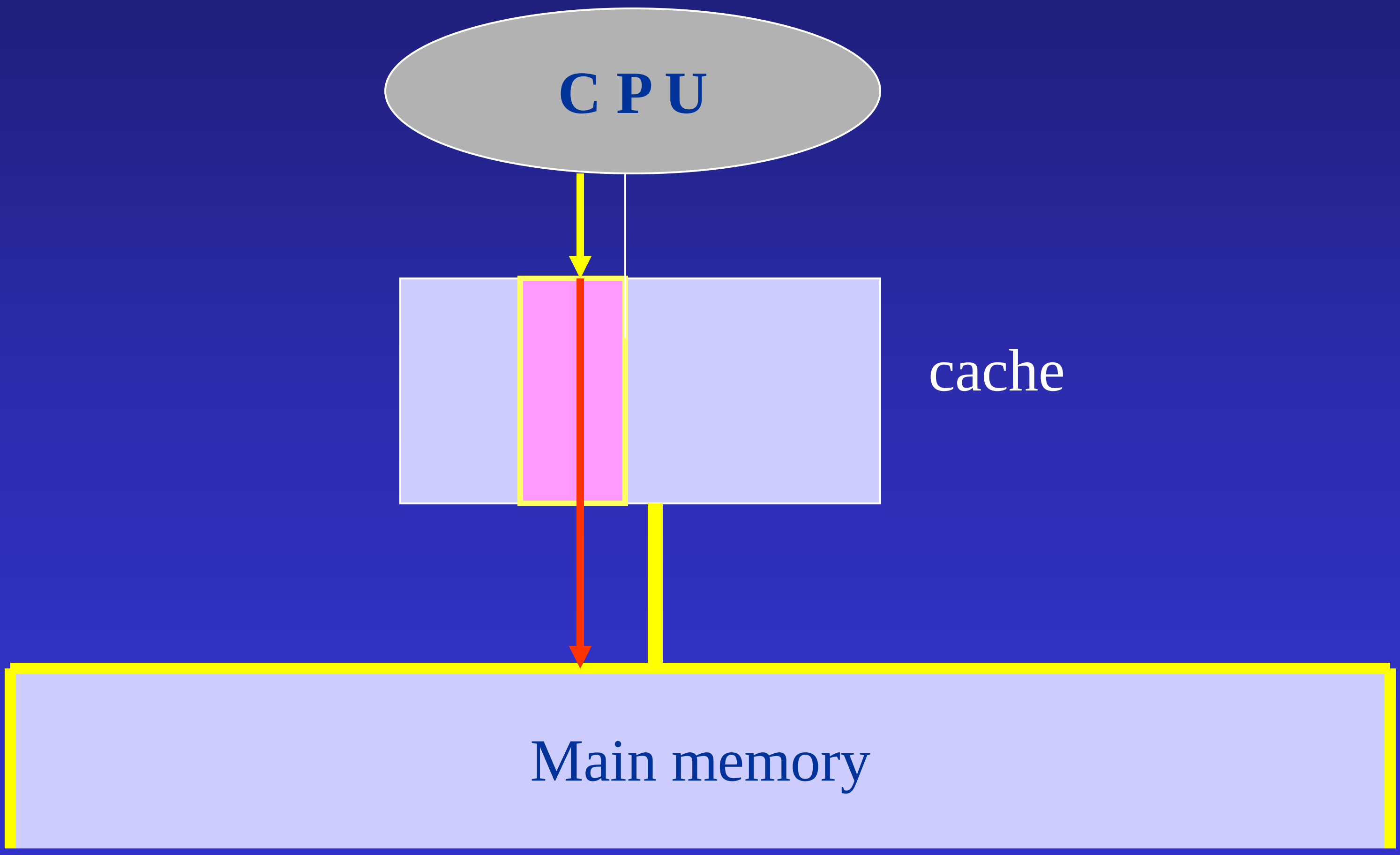
Dirty bit(s)

- Indicates if the block has been written to.
  - No need in I-caches.
  - No need in write through D-cache.
  - Write back D-cache needs it.

Write back



Write through



Cache Performance

Access time

t_{ea}=Hit time + Miss rate * Miss penalty

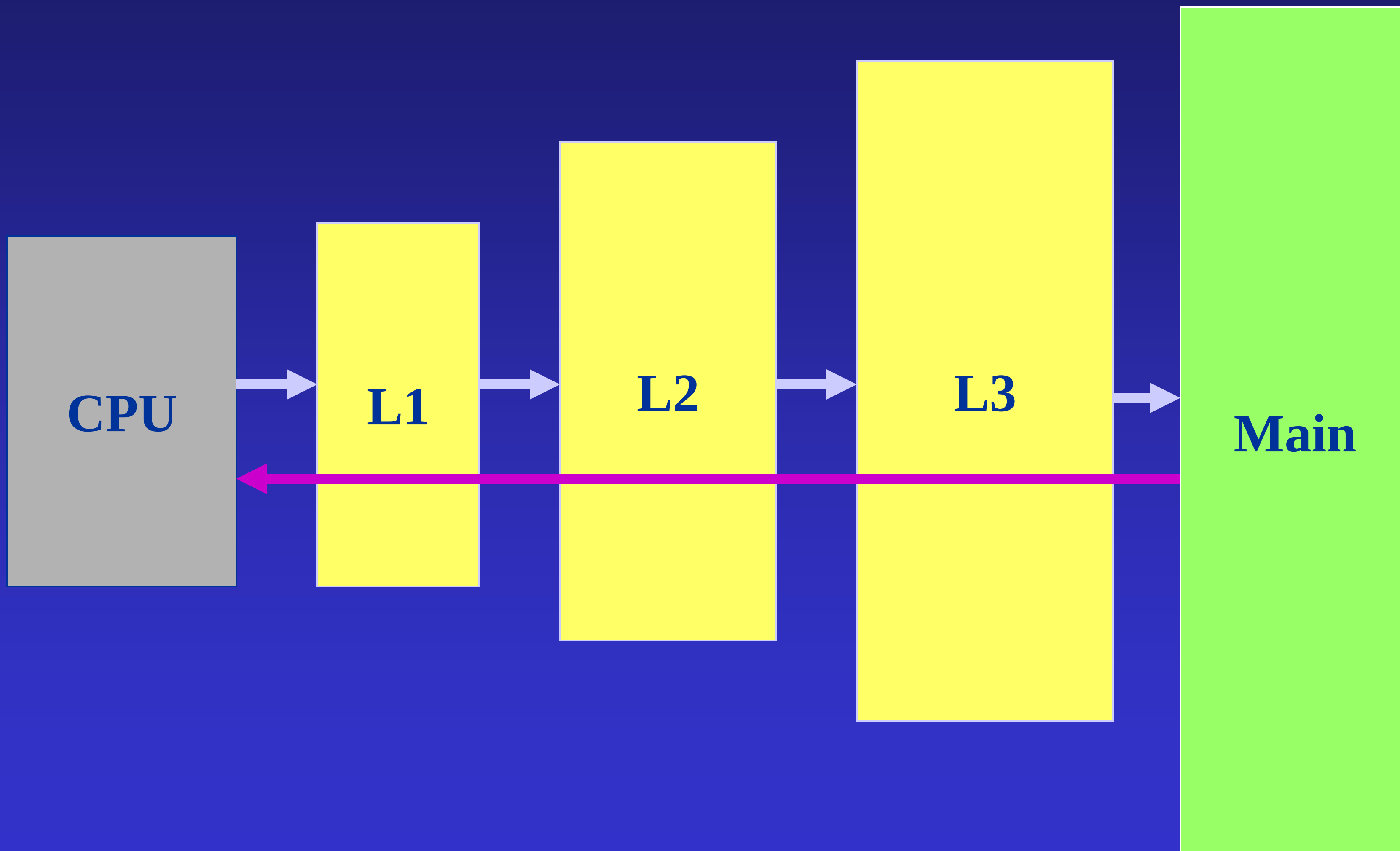
Associativity
(D-cache misses per 1000 instructions)

Size (KB)	2-way			4-way			8-way		
	LRU	Rdm	FIFO	LRU	Rdm	FIFO	LRU	Rdm	FIFO
16	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64	103.4	104.9	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

Reducing Cache Miss Penalties

- Multilevel cache
- Critical word first
- Priority to Read over Write
- Victim cache

Multilevel caches



Multilevel caches

- Average access time

$=$ Hit time$_{L1}$ + Miss rate$_{L1}$ * Miss penalty$_{L1}$

Hit time$_{L2}$ + Miss rate$_{L2}$ * Miss penalty$_{L2}$

Critical word first

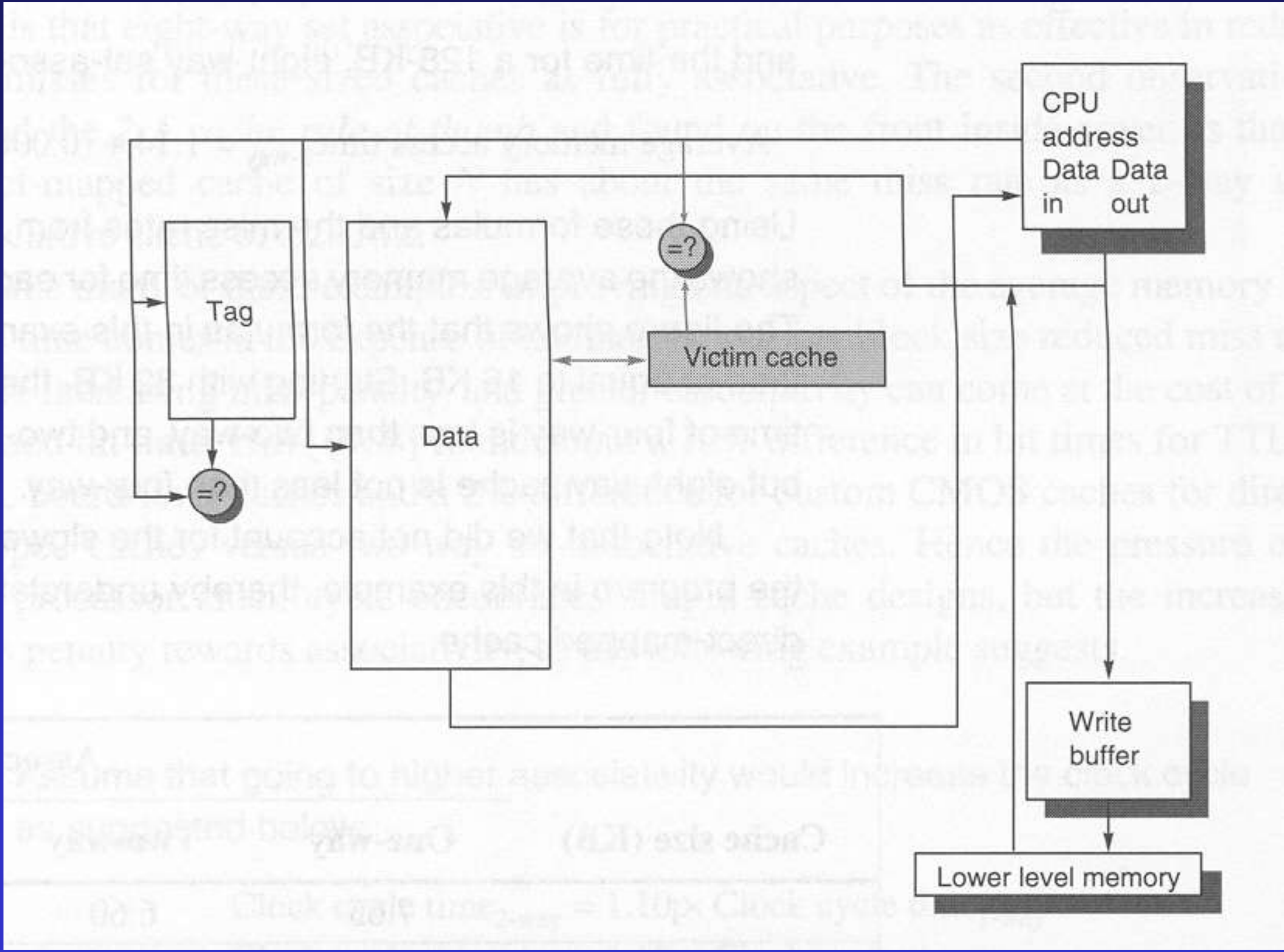
- The missed word is requested to be sent first from next level.

Priority to Read over Write misses

- Writes are not as critical as reads.
- CPU cannot continue if data or instruction is not read.

Victim cache

- Small fully associative cache.
- Contains blocks that have been discarded ("victims")
- Four entry cache removed 20-90% of conflict misses (4KB direct-mapped).


CPU
address
Data in
Data out
Tag
Data
=?
=?
Victim cache
Write buffer
Lower level memory

Pseudo Associative Cache

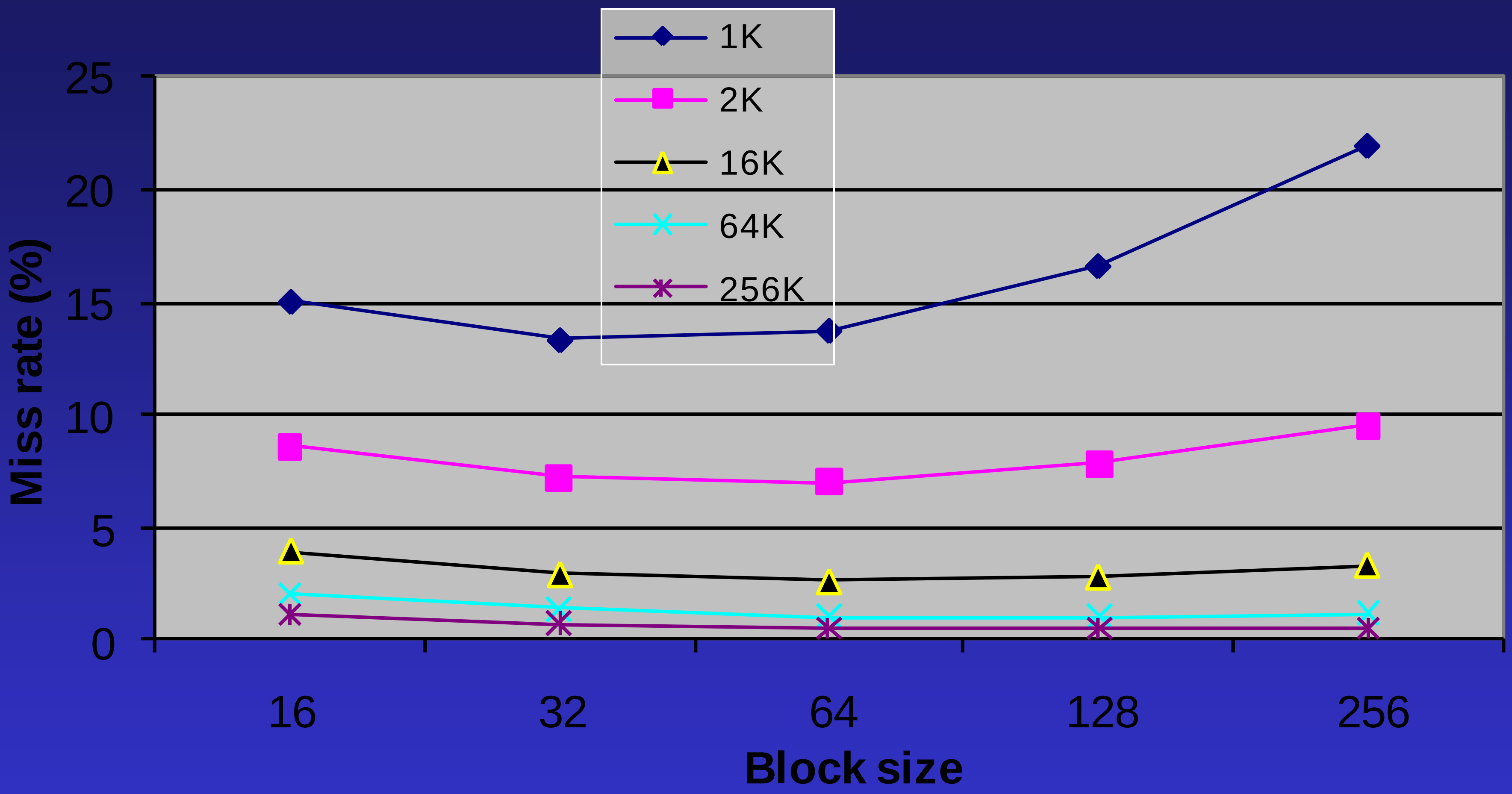
- Direct-mapped cache hit time
- If miss
  - Check other entry in cache
  - (change the most significant bit)
- If found no long wait

Reducing Cache Miss Rate

- Larger block size
- Larger caches
- Higher Associativity
- Pseudo-associative cache
- Prefetching

Larger Block Size

- Large block take advantage of spatial locality.
- On the other hand, less blocks in cache



1K
2K
16K
64K
256K
25
20
15
10
5
0
Miss rate (%)
16
32
64
128
256
Block size

Miss rate versus block size

Block size	Cache size			
	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

Example

- Memory takes 80 clock cycles of overhead
- Delivers 16 bytes every 2 clock cycles (cc)

Mem. System supplies	16 bytes in 82 cc
	32 bytes in 84 cc

Example (cont'd)

Average access time

$= \text{Hit time}_{L1} + \text{Miss rate}_{L1} * \text{Miss penalty}_{L1}$

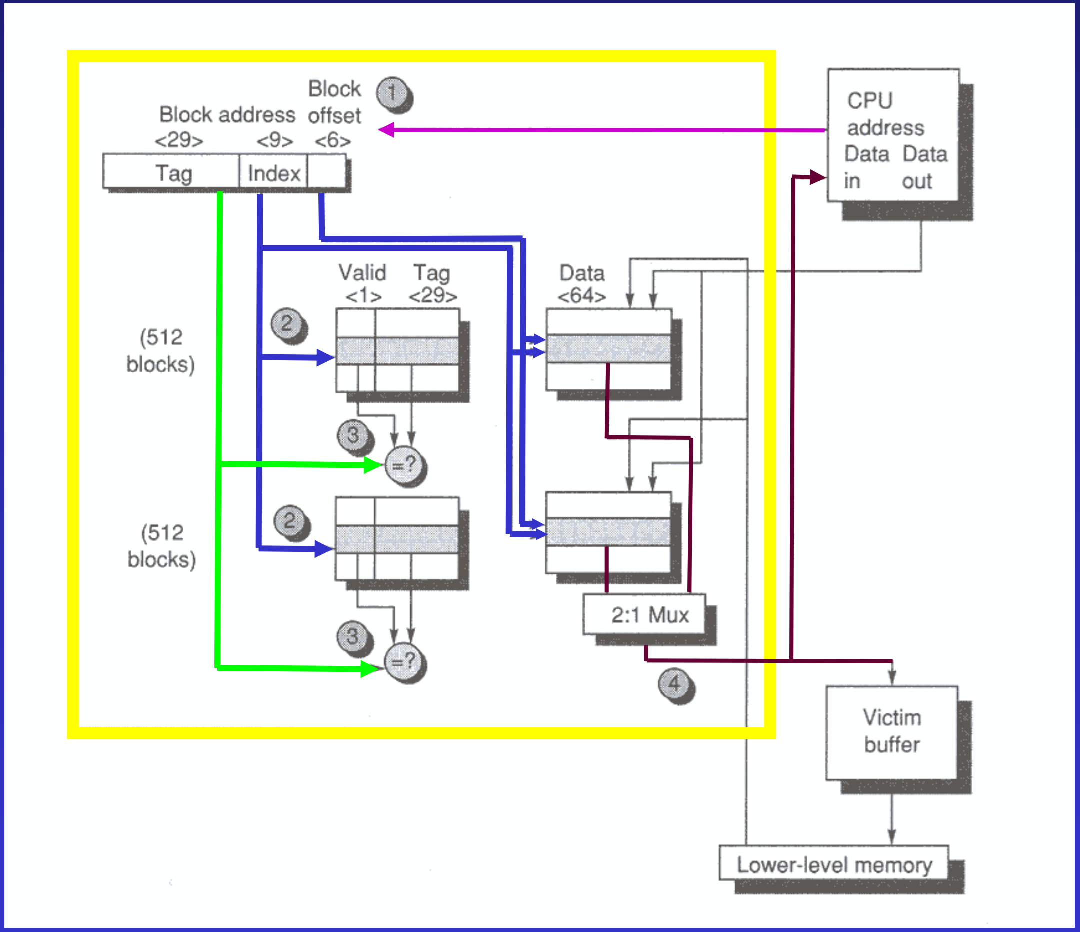
For a 16-byte block in a 4 KB cache

Ave. access time = 1 + (8.57% * 82)=8.0274

Ave. mem. access time versus block size

Block size	Miss penalty	Cache size			
		4K	16K	64K	256K
16	82	8.027	4.231	2.673	1.894
32	84	**7.082**	3.411	2.134	1.588
64	88	7.160	**3.323**	**1.933**	**1.449**
128	96	8.469	3.659	1.979	1.470
256	112	11.651	4.685	2.288	1.549

Two-way cache (Alpha)



Higher Associativity

- Having higher associativity is NOT for free
- Slower clock may be required
- Thus, there is a trade-off between associativity (with higher hit rate) and faster clocks (direct-mapped).

Associativity example

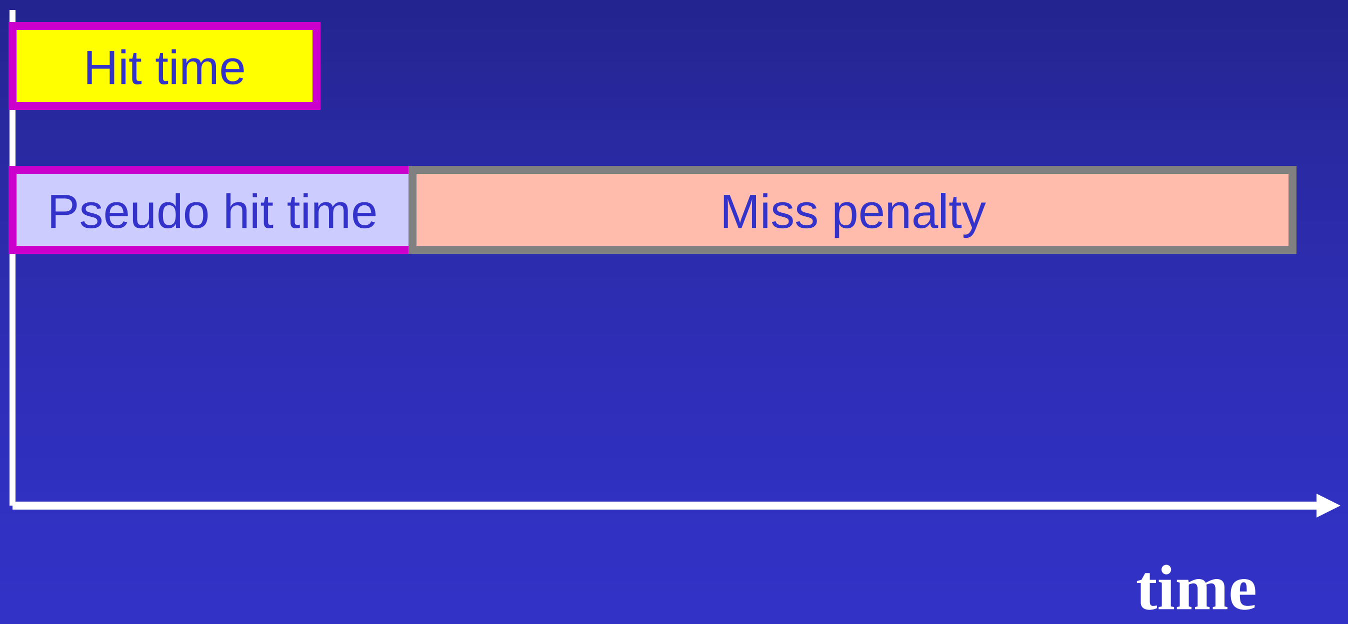
- If clock cycle time (cct) increases as follows:
  - $cct_{2\text{-way}} = 1.1cct_{1\text{-way}}$
  - $cct_{4\text{-way}} = 1.12cct_{1\text{-way}}$
  - $cct_{8\text{-way}} = 1.14cct_{1\text{-way}}$
- Miss penalty is 50 cycles
- Hit time is 1 cycle

Cache size (KB)	Associativity			
	One-way	Two-way	Four-way	Eight-way
1	7.65	6.60	6.22	5.44
2	5.90	4.90	4.62	4.09
4	4.60	3.95	3.57	3.19
8	3.30	3.00	2.87	2.59
16	2.45	2.20	2.12	2.04
32	2.00	1.80	1.77	**1.79**
64	1.70	1.60	1.57	**1.59**
128	1.50	1.45	1.42	**1.44**

Pseudo Associative Cache

- Direct-mapped cache hit time
- If miss
  - Check other entry in cache
  - (change the most significant bit)
- If found no long wait

Pseudo Associative



Hardware prefetching

- What about fetching two blocks in a miss.
  - Alpha AXP 21064
- Problem: real misses may be delayed due to prefetching

Fetch Policies

- Memory references:
  - 90% reads
  - 10% writes
- Thus, read has higher priority

Fetch

- Fetch on miss
  - Demand fetching
- Prefetching
  - Instructions (I-Cache)

Hardware prefetching

- What about fetching two blocks in a miss.
  - Alpha AXP 21064
- Problem: real misses may be delayed due to prefetching

Compiler Controlled Prefetching

- Register prefetching
  - Load values in regs before they are needed
- Cache prefetching
  - Loads data in cache only

This is only possible if we have nonblocking or lockup-free caches.

Order Main Memory to Cache

- Needed AU is in the middle of the block
- Block load
  - The complete block is loaded
- Load forward
  - The needed AU is forwarded
  - AUs behind the miss are not loaded
- Fetch bypass
  - Start with needed AU and AUs behind are loaded later

Access time

$$\text{Mem\_ access\_time} = \text{Hit\_rate} + \text{Miss\_rate} \times \text{Miss\_penalty}$$

$$\text{Mem\_ access\_time}_{L1} = \text{Hit\_rate}_{L1} + \text{Miss\_rate}_{L1} \times \text{Miss\_penalty}_{L1}$$

$$\text{Miss\_penalty}_{L1} = \text{Hit\_rate}_{L2} + \text{Miss\_rate}_{L2} \times \text{Miss\_penalty}_{L2}$$

Reducing Hit Time

- Critical in increasing the processor clock frequency.

Small & simple caches

- "smaller" hardware is faster
  - Less sequential operations
- Direct mapping is simpler

Avoid address translation

- Virtual address → Physical address.