ARCOS Group

uc3m | Universidad Carlos III de Madrid

Lesson 5 (II)
Memory hierarchy

Computer Structure

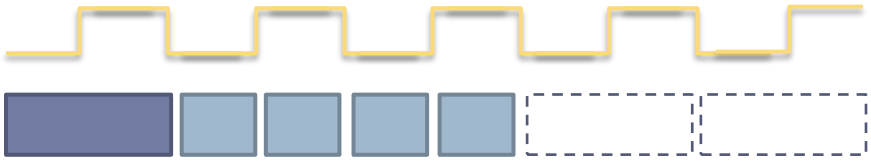
Bachelor in Computer Science and Engineering

Contents

Main memory characteristics

- It is better to access to consecutive words
  - Example 1: access to 5 individuals non-consecutives words

  - Example 2: access to 5 consecutives words



ARCOS @ UC3M
Félix García Carballeira, Alejandro Calderón Mateos

Characteristics of memory accesses

- "Principle of proximity or locality of references":

  During the execution of a program, references (addresses) to memory tend to be grouped by:

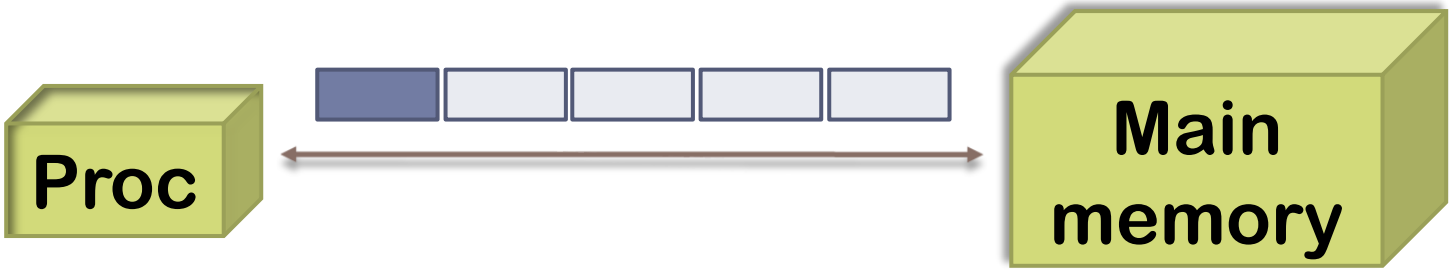
  - Spatial proximity
    - Sequence of instructions
    - Sequential access to arrays
  - Temporal proximity
    - loops

```
.data
  vector: .space 4*1024

.text
 main:      li   t0 0
            la   t1 vector
            li   t3 1024
            li   t4 4
       b2:  bge  t0 t3 fb2
            mul  t2 t0 t4
            add  t2 t1 t2
            sw   t0 0(t2)
            addi t0 t0 1
            beq  x0 x0 b2
       fb2: jr ra
```

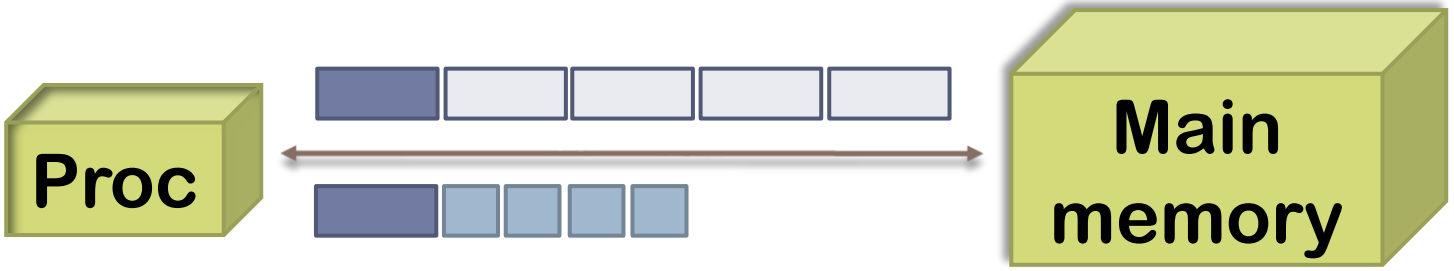
Goal of the cache memory: to take advantage of contiguous accesses

- If when accessing a memory location only the data of that location is transferred, possible accesses to contiguous data are not taken advantage of.



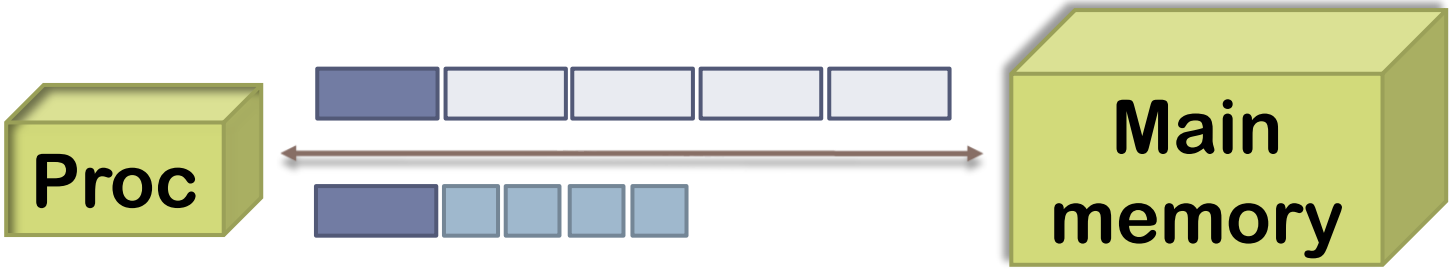
Goal of the cache memory: to take advantage of contiguous accesses

- If, when accessing a memory location, this data and the contiguous data are transferred, the access to contiguous data is exploited



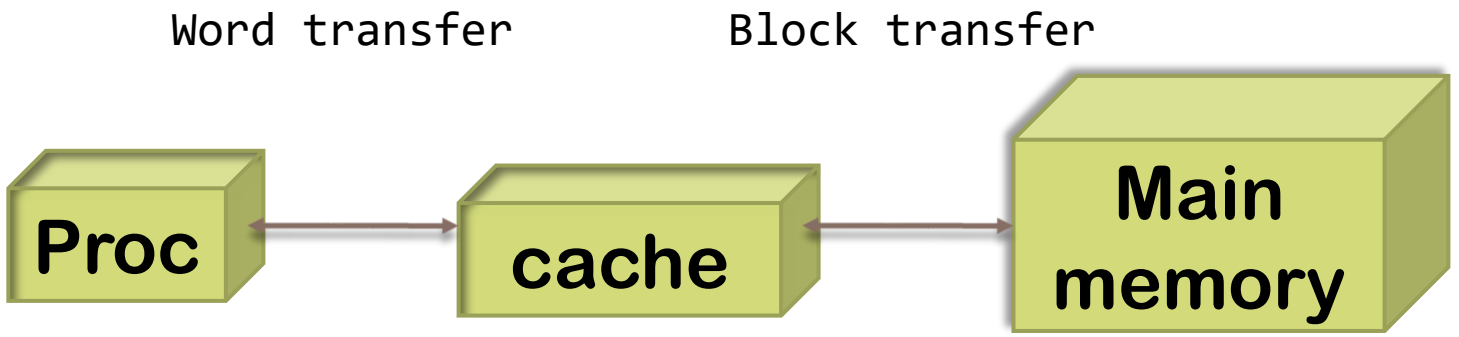
Goal of the cache memory: to take advantage of contiguous accesses

- If, when accessing a memory location, this data and the contiguous data are transferred, the access to contiguous data is exploited
  - I transfer from the main memory a block of words
  - Where are the words of the block stored?

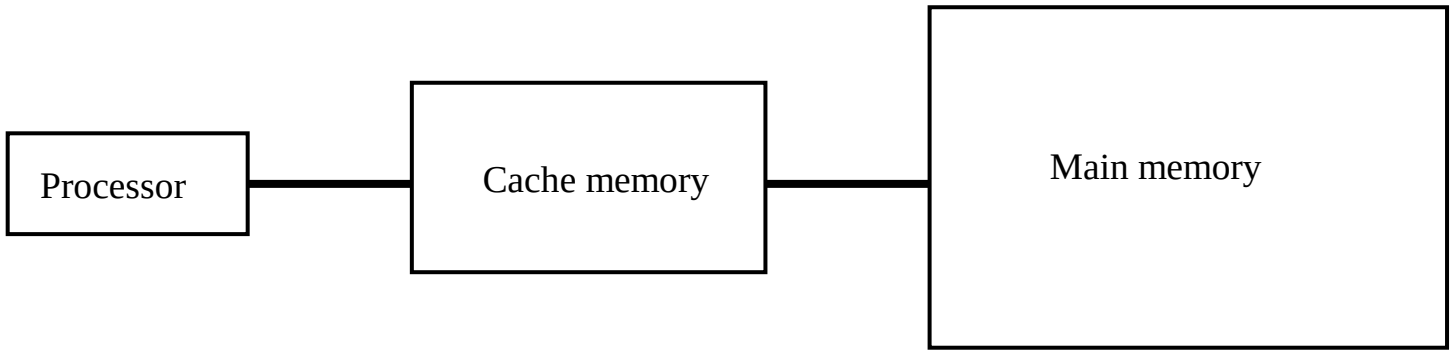


Cache memory

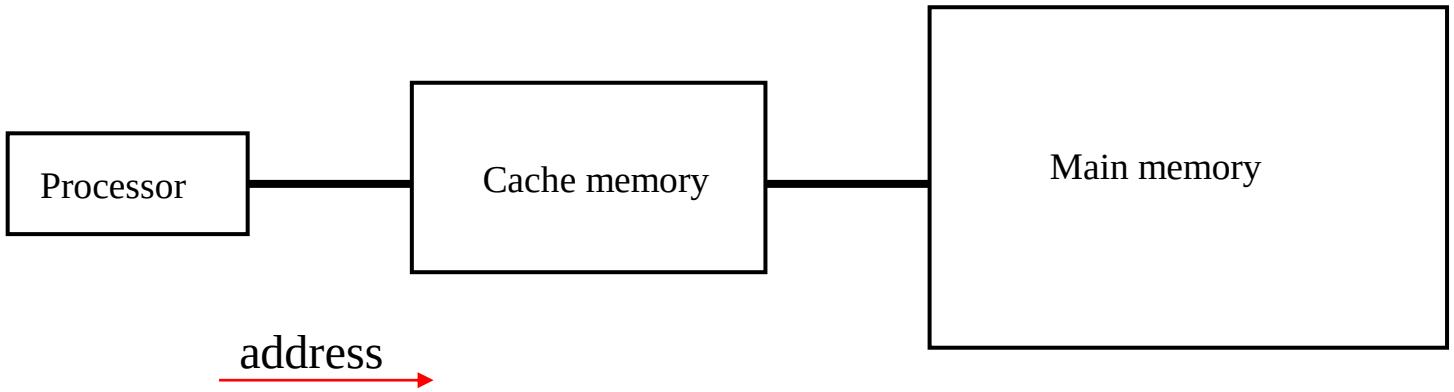
- Small amount of fast SRAM memory
  - Integrated in the Processor itself
  - Faster and more expensive than the DRAM
- Between main memory and processor
- Stores a copy of chunks of the main memory



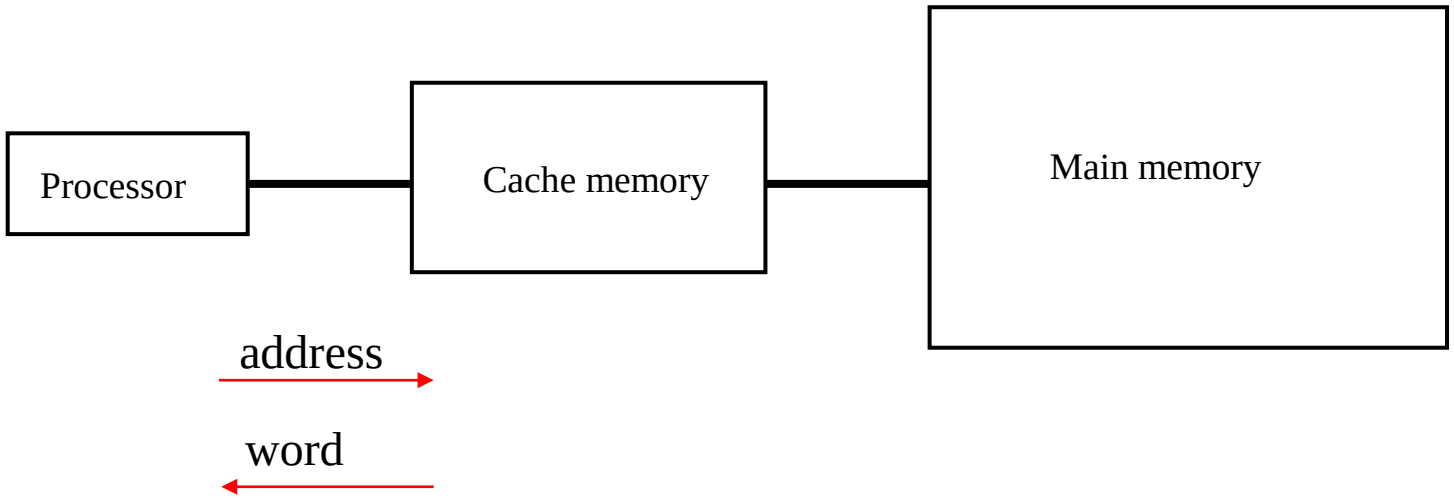
How cache memory works



How cache memory works

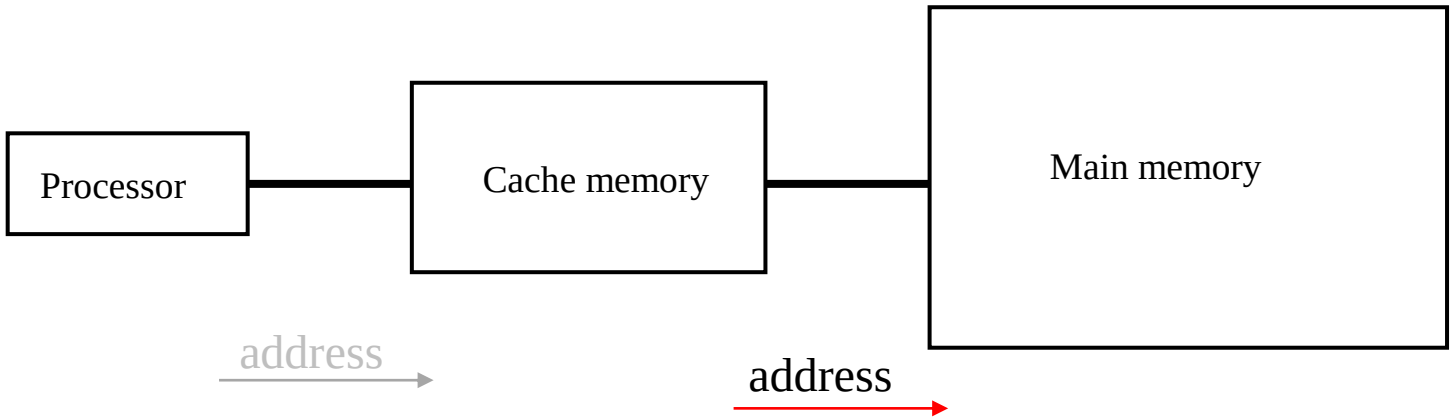


How cache memory works



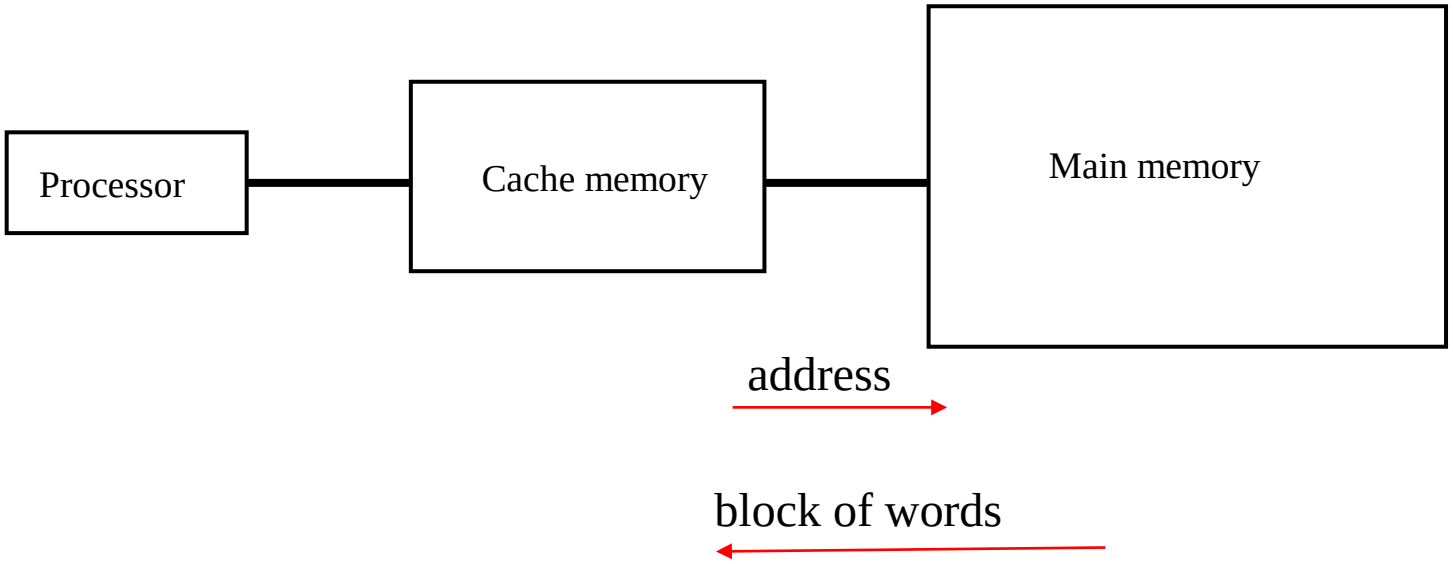
HIT

How cache memory works

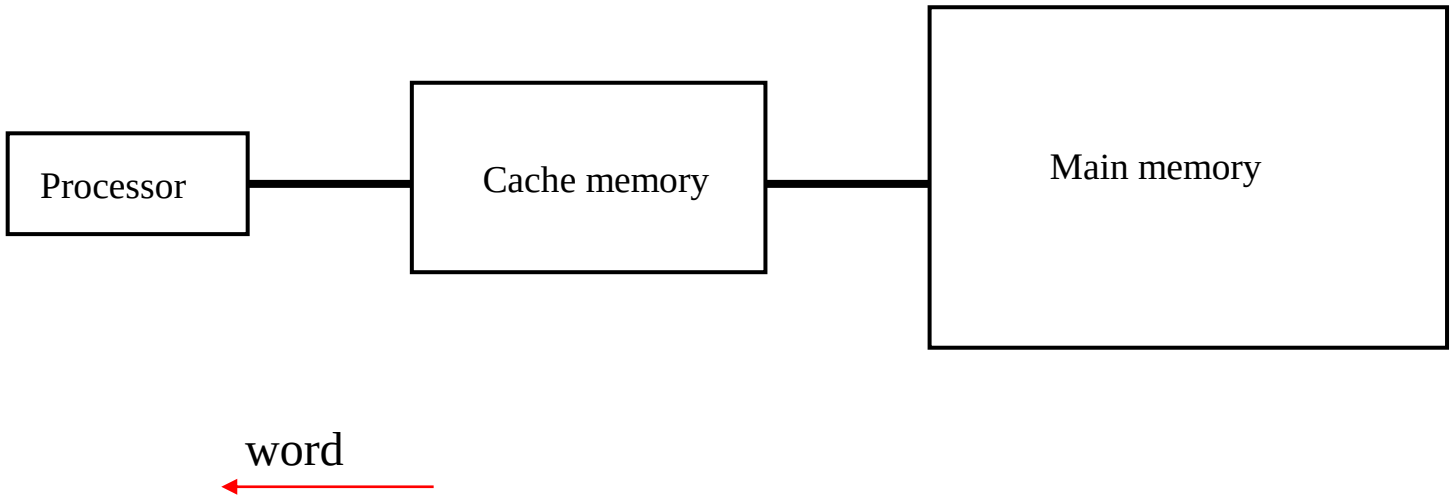


MISS

How cache memory works

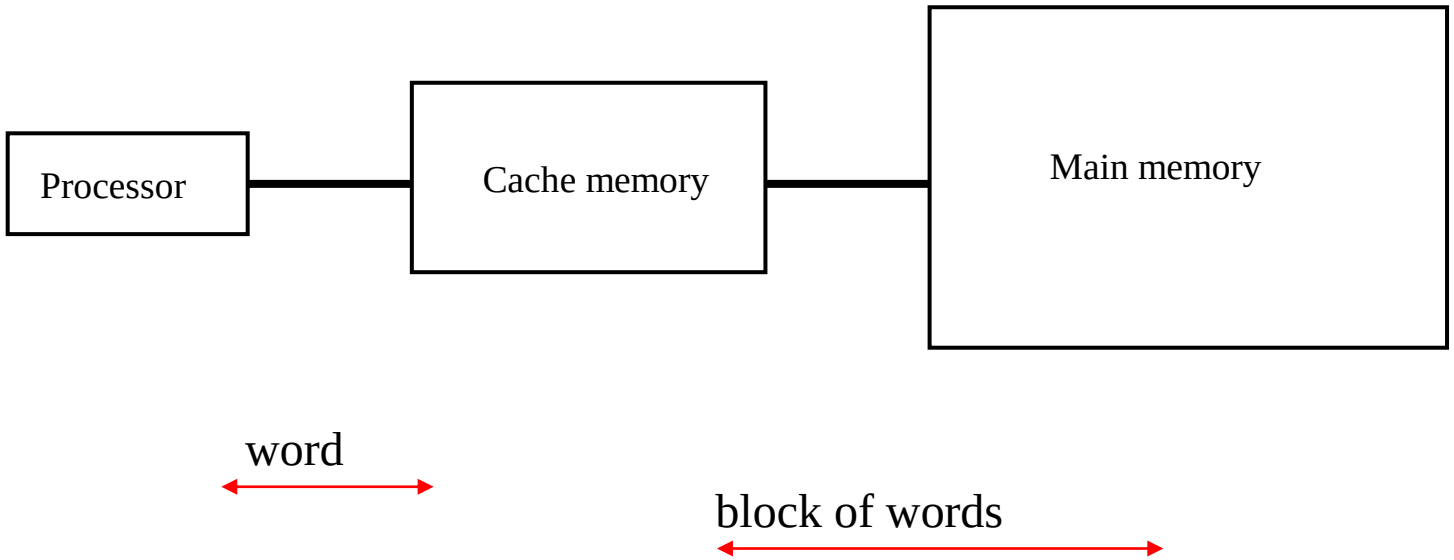


How cache memory works

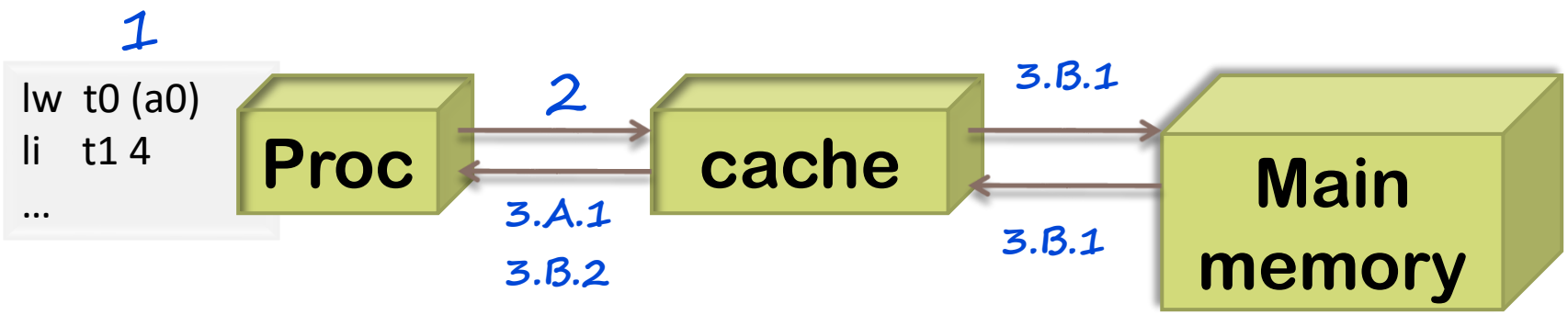


How cache memory works

summary



How it works (in general)



1. The Processor performs an access to cache.
2. The cache checks if the data for this position is already there:
   - **If it is there (HIT),**
     - **3.A.1** It is served to the Processor from the cache (quickly): Ta
   - **If it is not there (MISS),**
     - **3.B.1** The cache transfers from Main memory the block associated with position: Tf
     - **3.B.2** The cache then delivers the requested data to the processor: Ta

Example of how it works

```
int i;
int s = 0;
for (i=0; i < 1000; i++)
    s = s + i;
```

```
        li   t0, 0     # s
        li   t1, 0     # i
        li   t2, 1000
bucle:  bge  t1, t2, fin
        add  t0, t0, t1
        addi t1, t1, 1
        beq  x0, x0, bucle
fin:    …
```

- Example:
  - Cache access: 2 ns
  - Main memory Access: 120 ns
  - Cache block: 4 words
  - Transfer a block between main memory and cache: 200 ns

Example of how it works

```
int i;
int s = 0;
for (i=0; i < 1000; i++)
    s = s + i;
```

```
        li   t0, 0     # s
        li   t1, 0     # i
        li   t2, 1000
bucle:  bge  t1, t2, fin
        add  t0, t0, t1
        addi t1, t1, 1
        beq  x0, x0, bucle
fin:    …
```

- Without cache memory:
  - Number of memory access = 3 + 4 × 1000 + 1 = 4004 access
  - Total access time = 4004 × 120 = 480480 ns = 480480 ms

Example of how it works

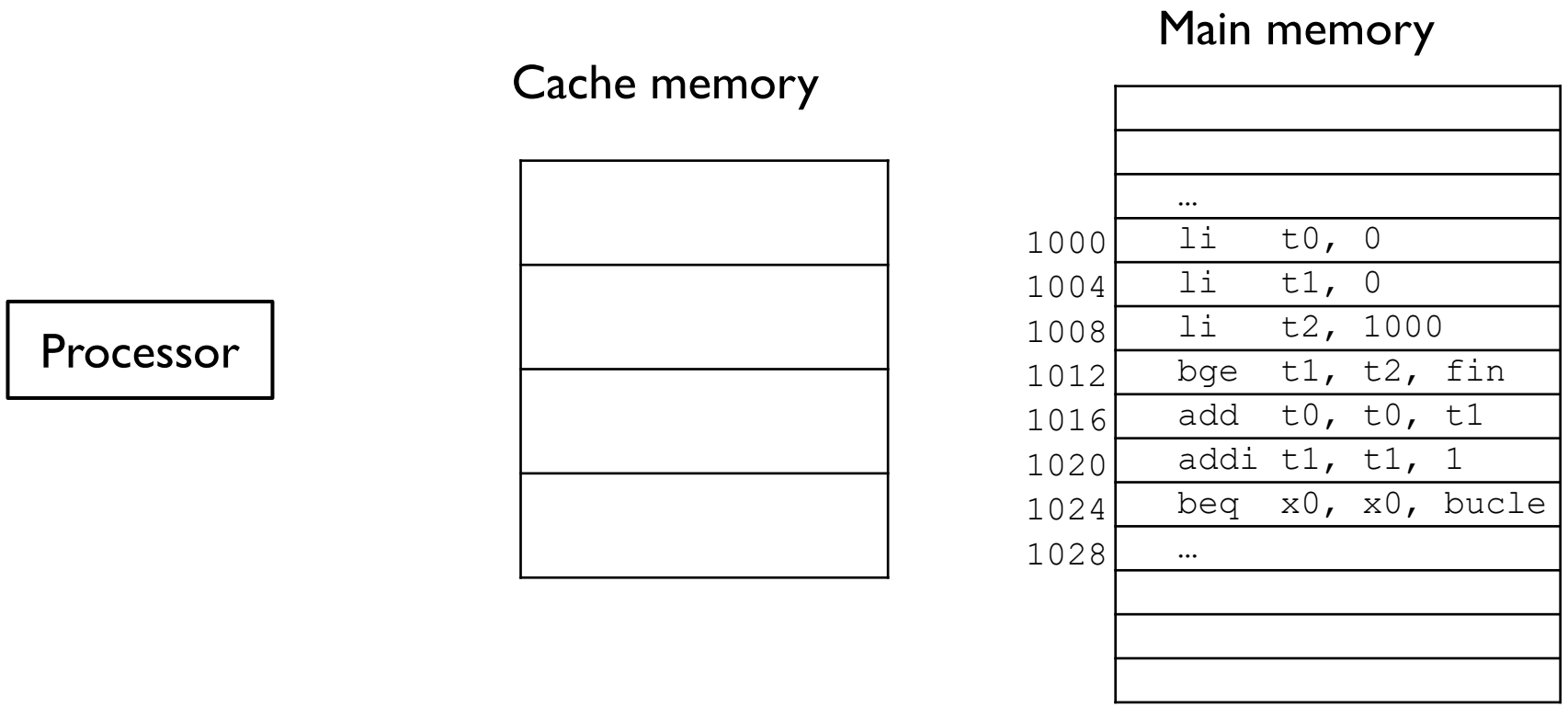
```
int i;
int s = 0;
for (i=0; i < 1000; i++)
    s = s + i;
```

```
        li   t0, 0     # s
        li   t1, 0     # i
        li   t2, 1000
bucle:  bge  t1, t2, fin
        add  t0, t0, t1
        addi t1, t1, 1
        beq  x0, x0, bucle
fin:    …
```

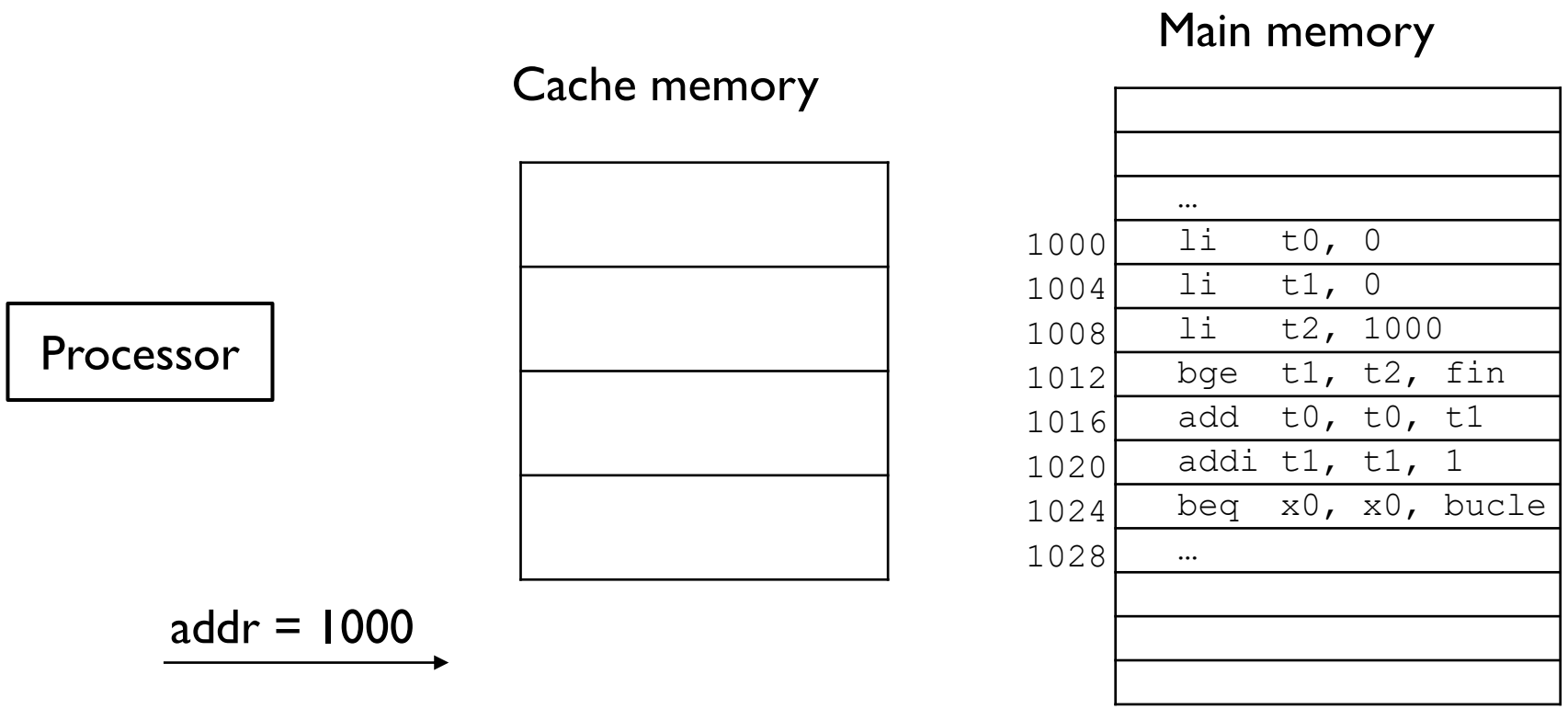
- With cache memory (blocks of 4 words):

ARCOS @ UC3M
Félix García Carballeira, Alejandro Calderón Mateos

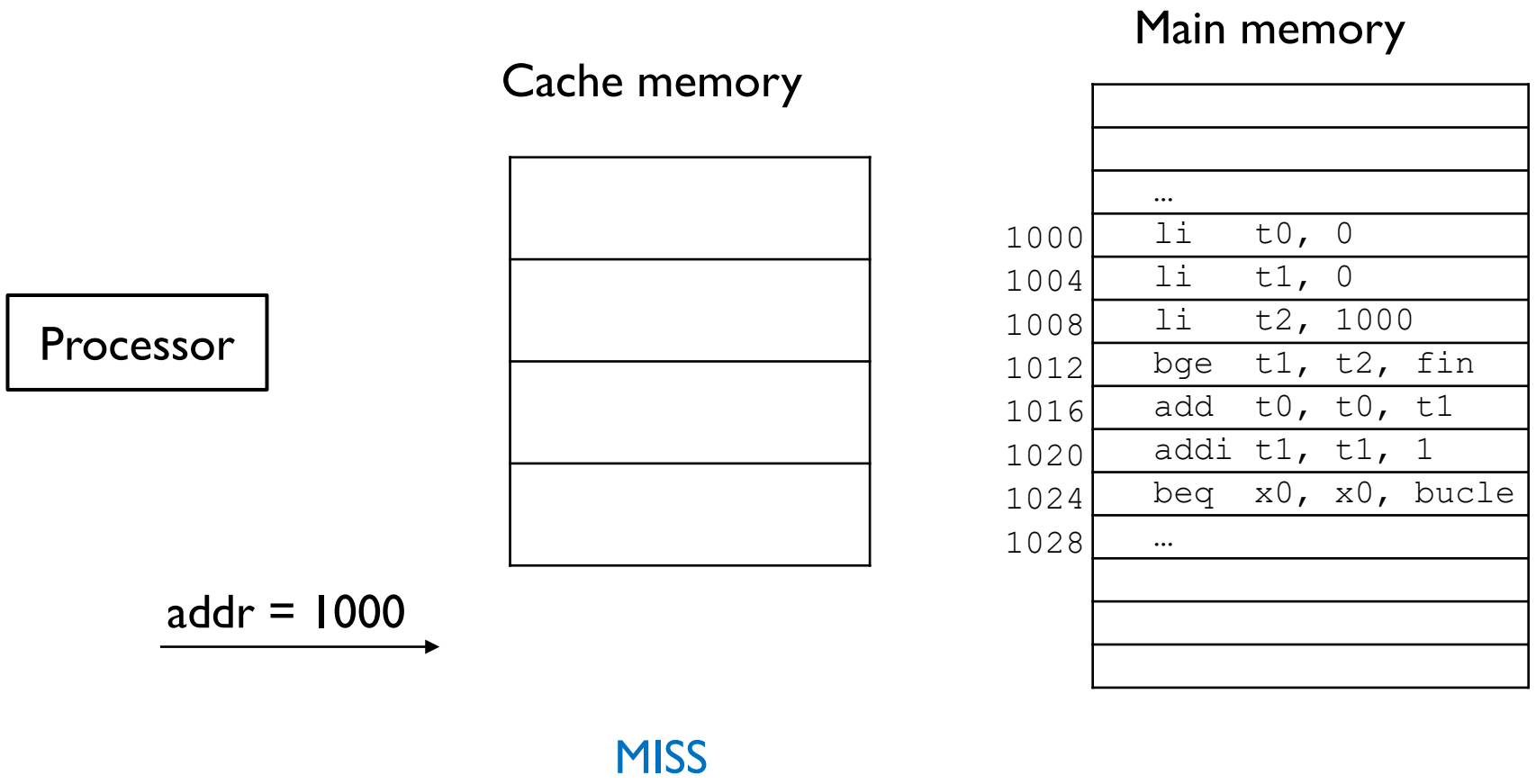
Example of how it works



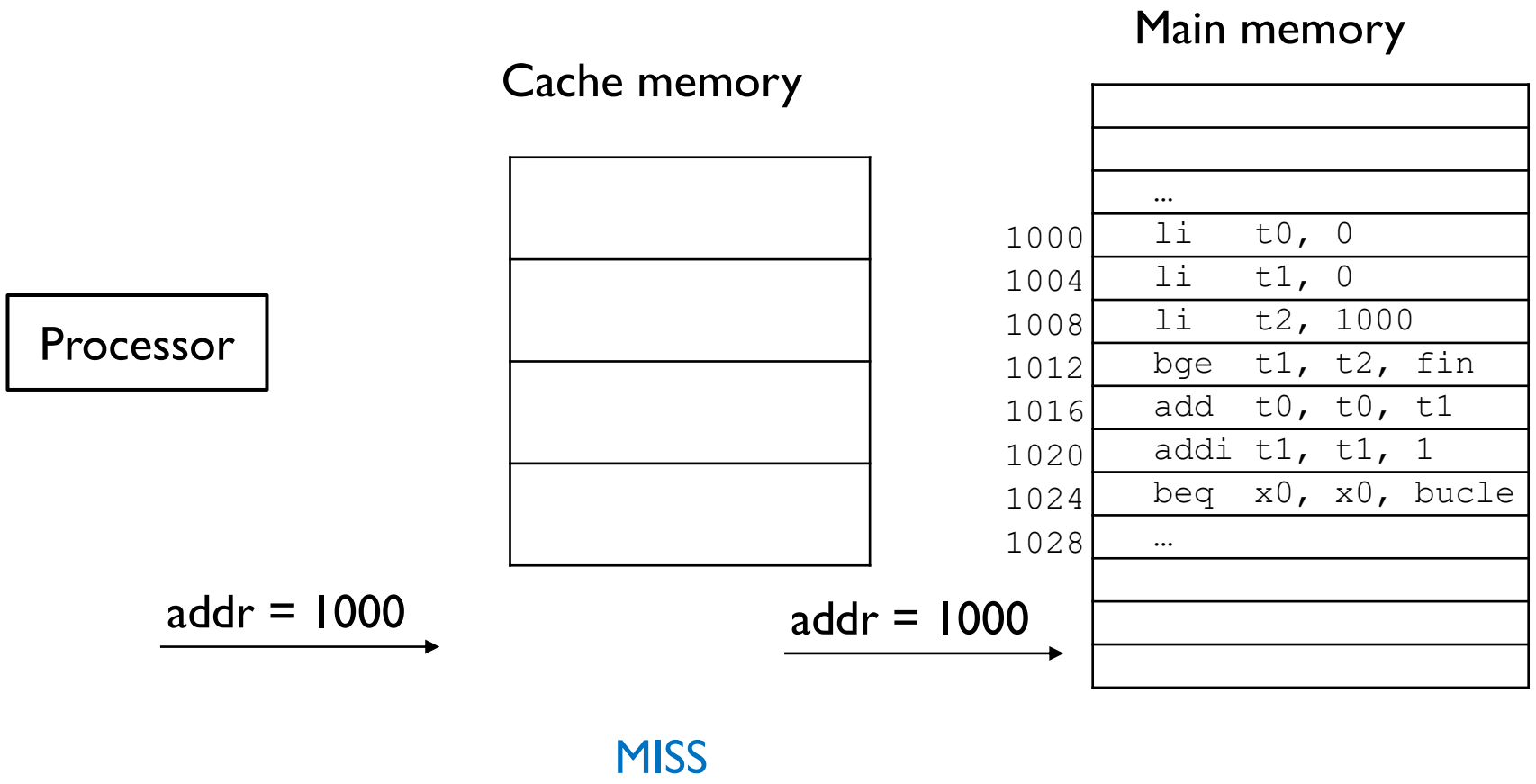
Example of how it works



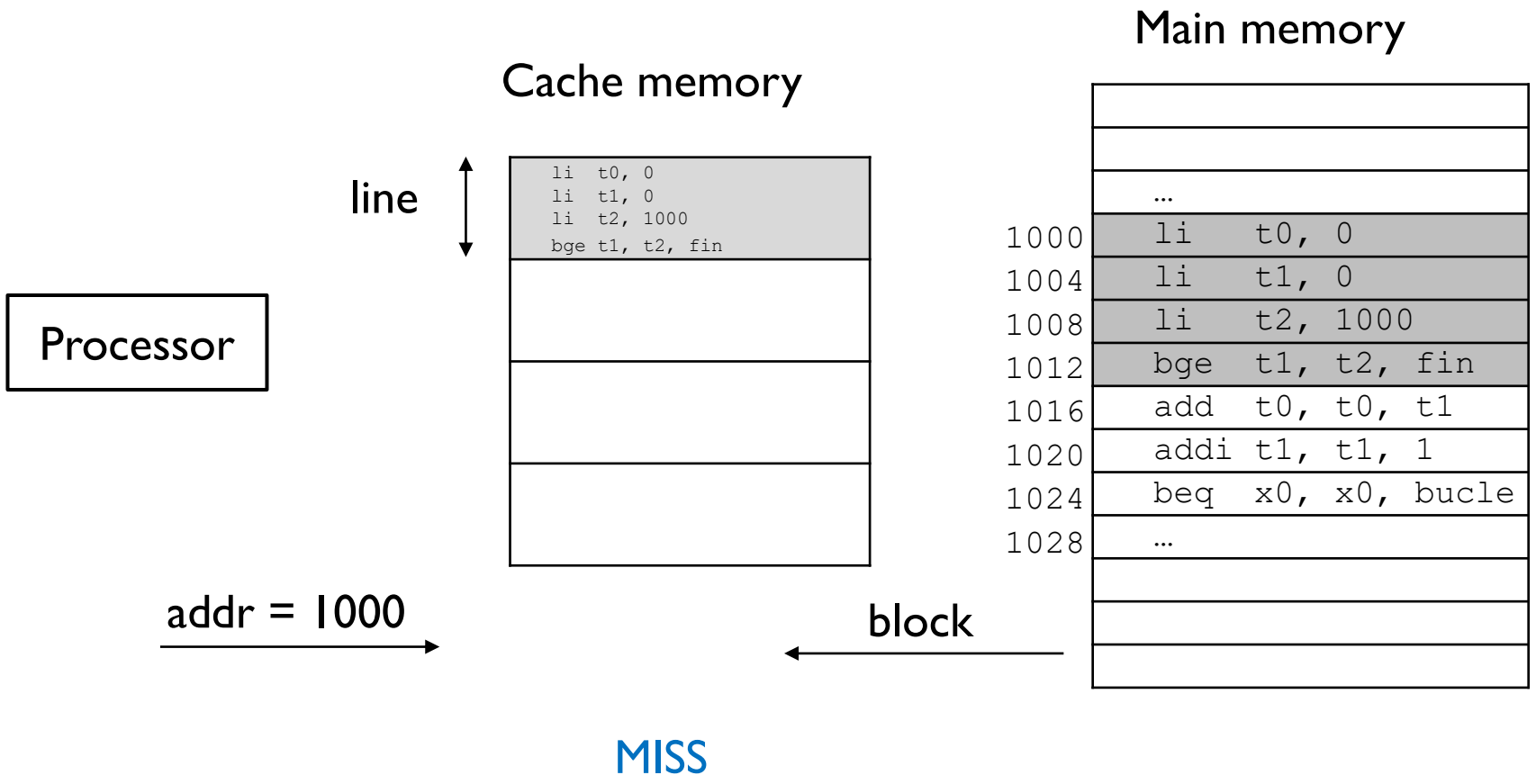
Example of how it works



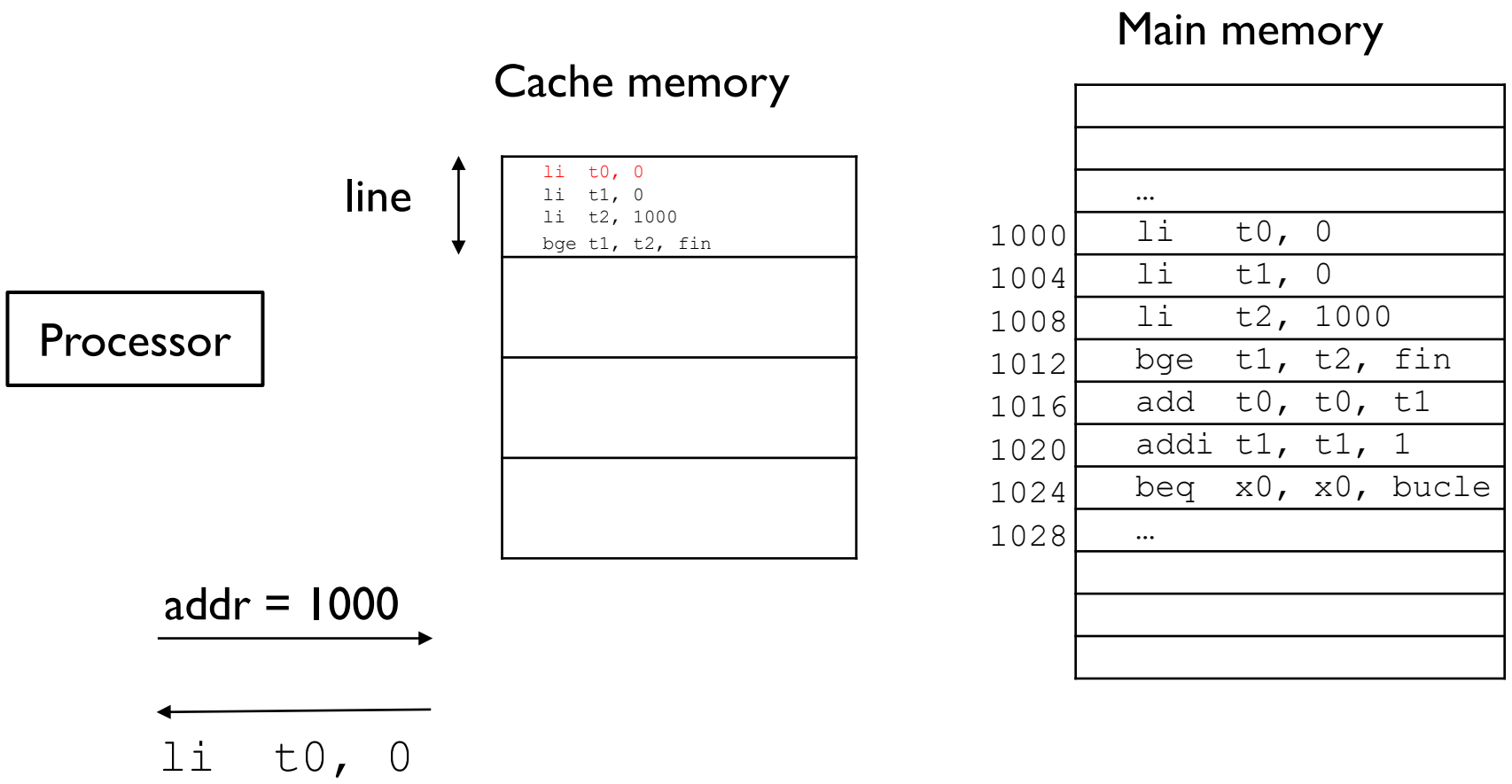
Example of how it works



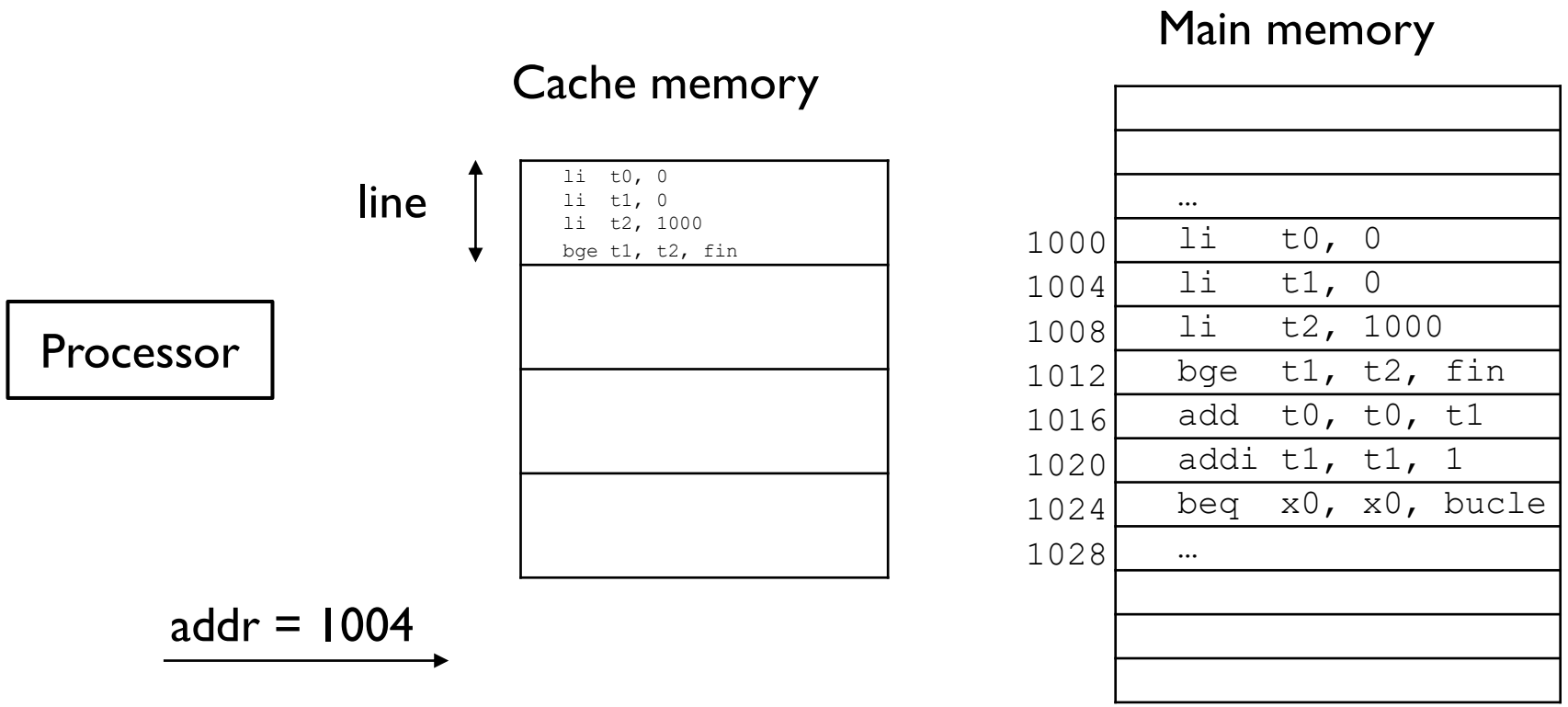
Example of how it works



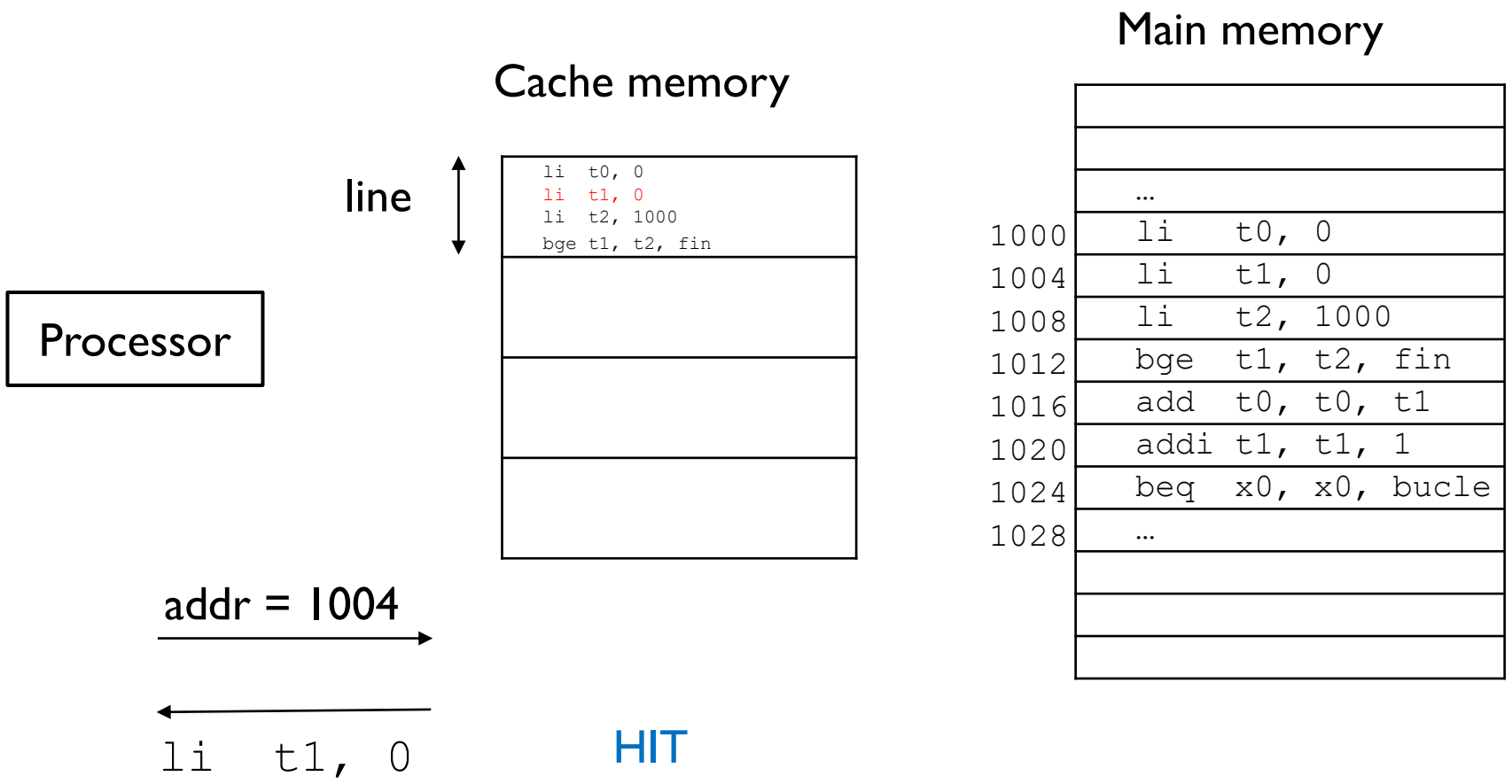
Example of how it works



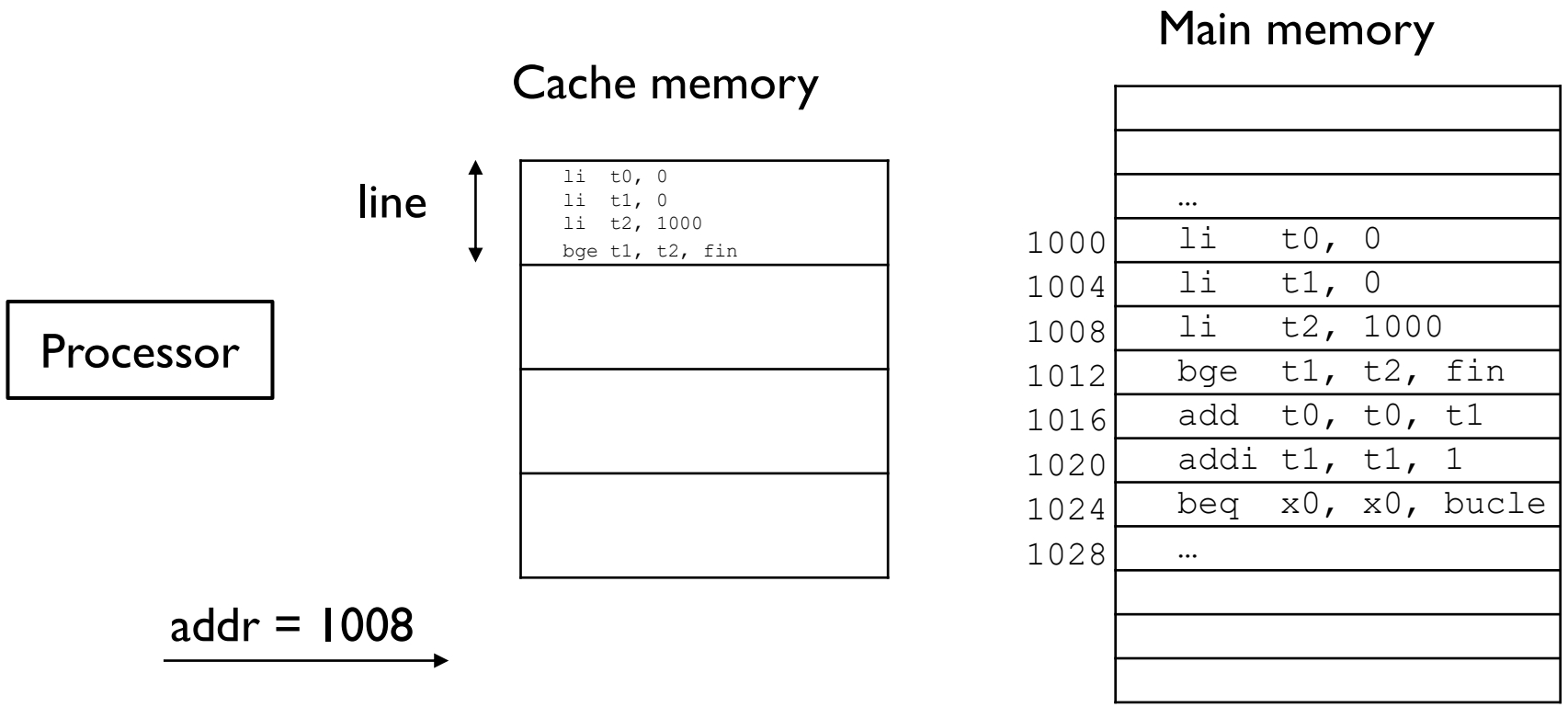
Example of how it works



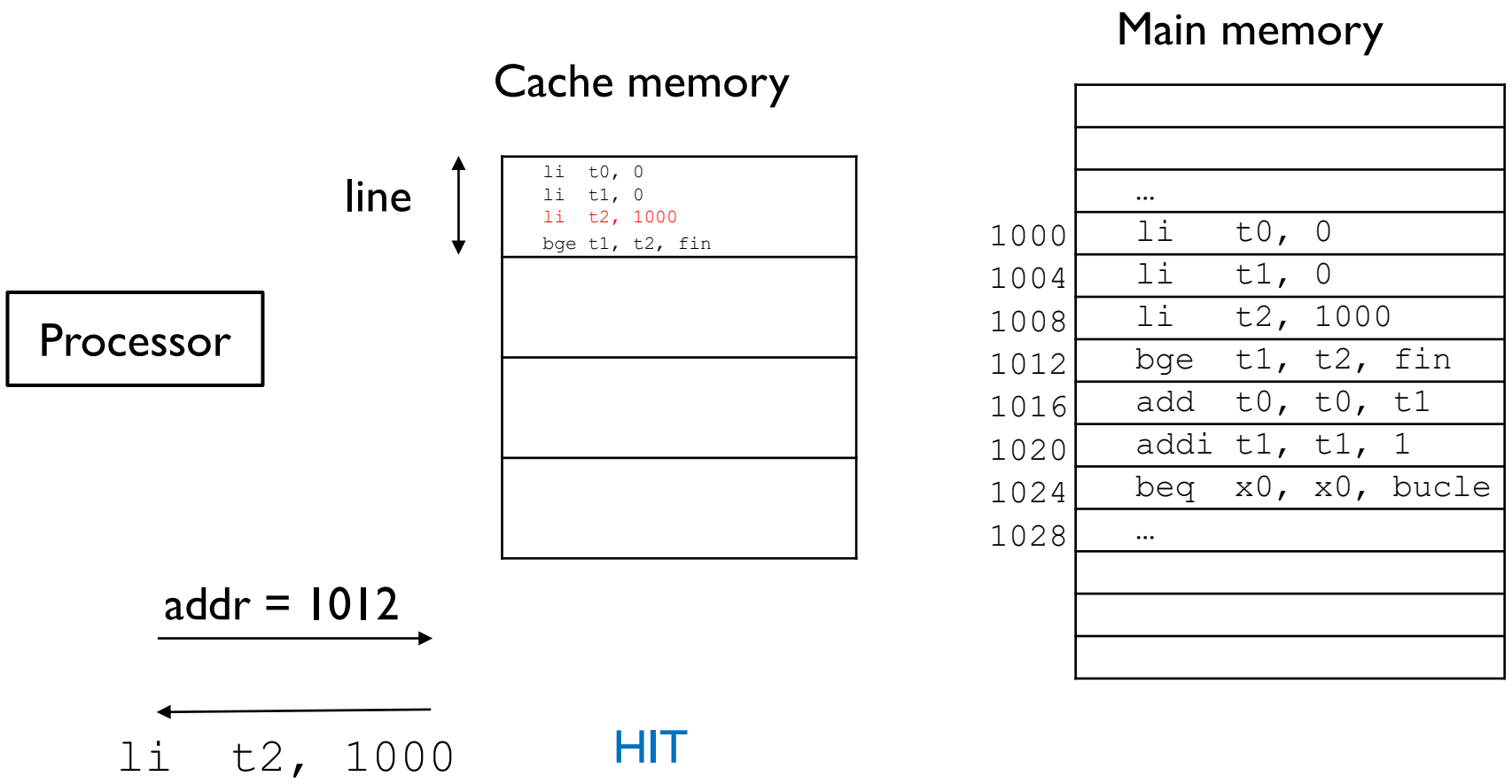
Example of how it works



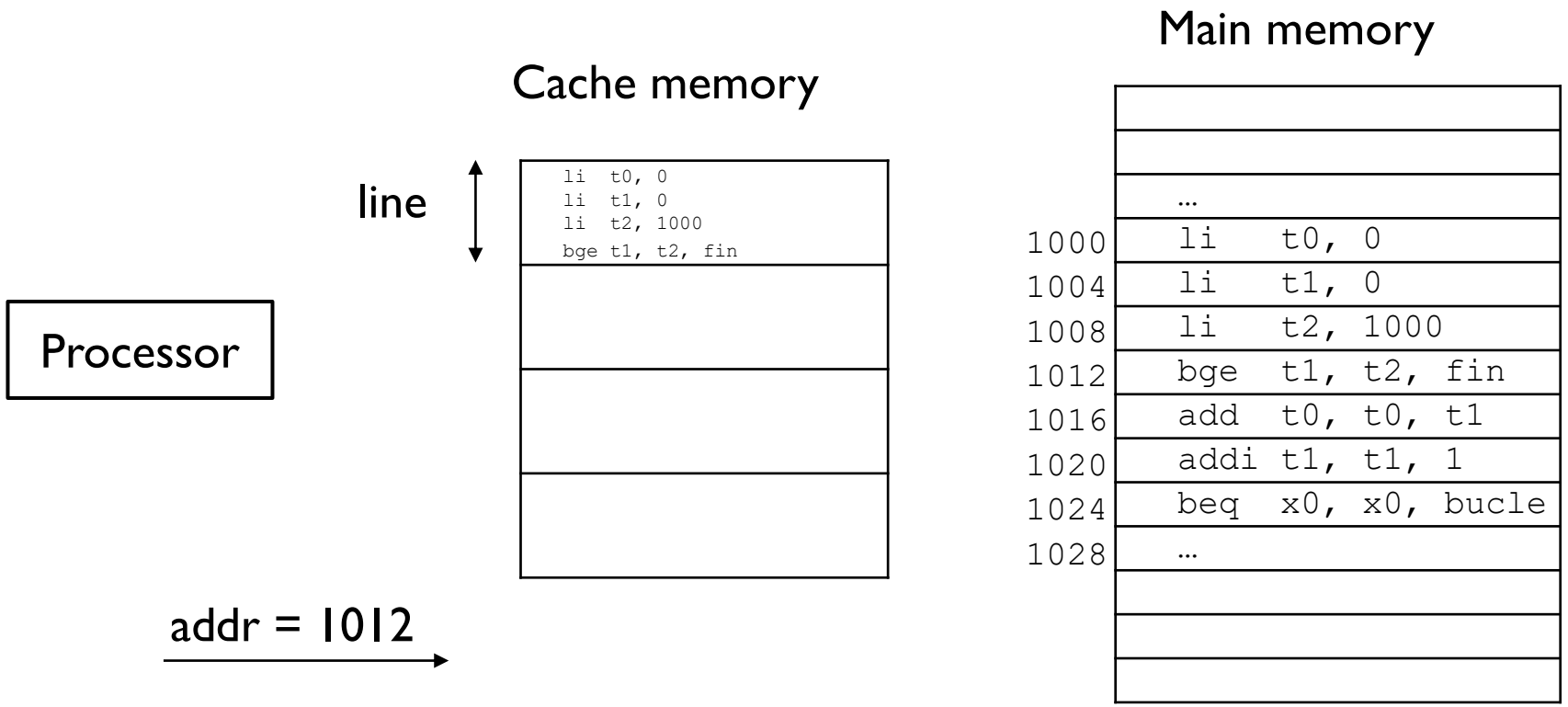
Example of how it works



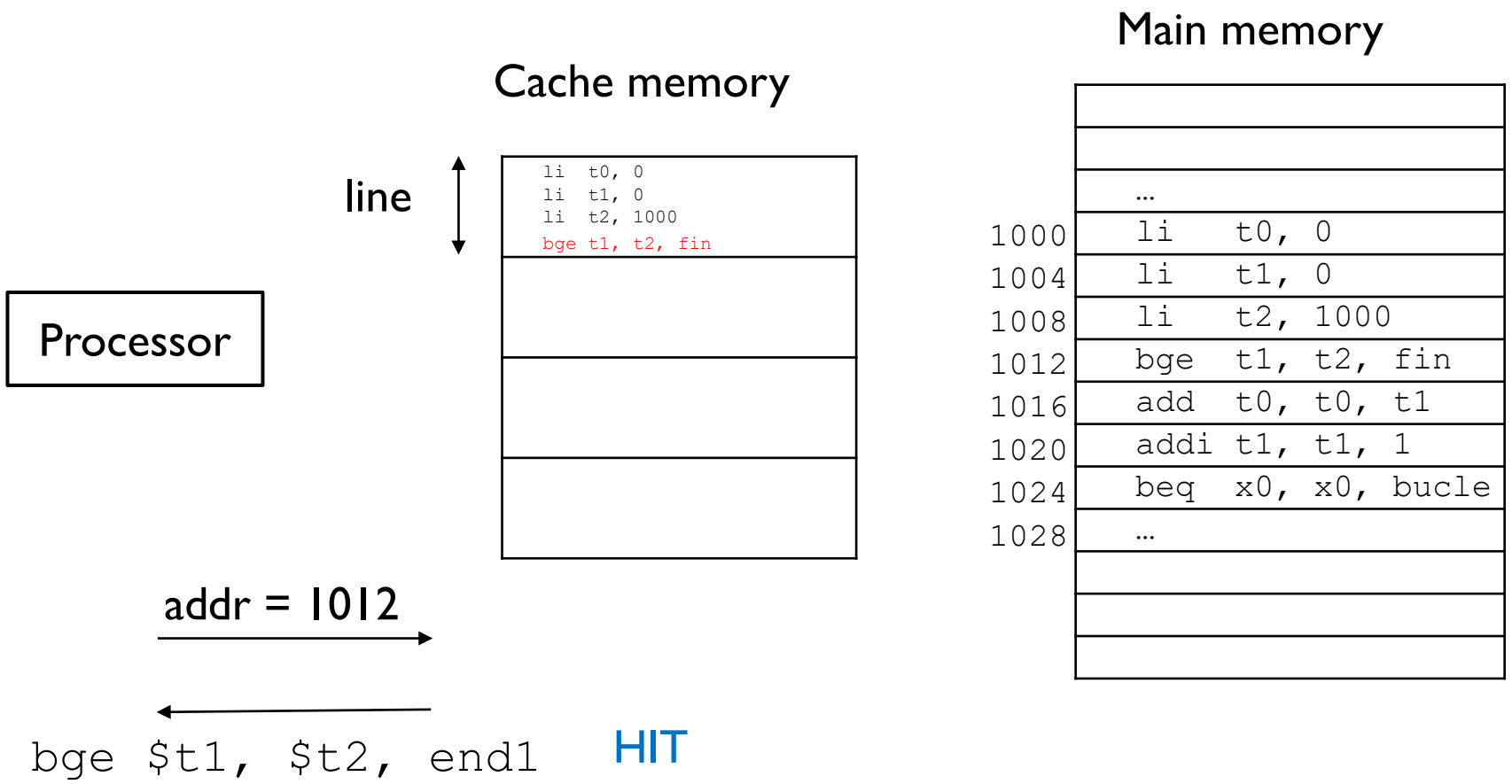
Example of how it works



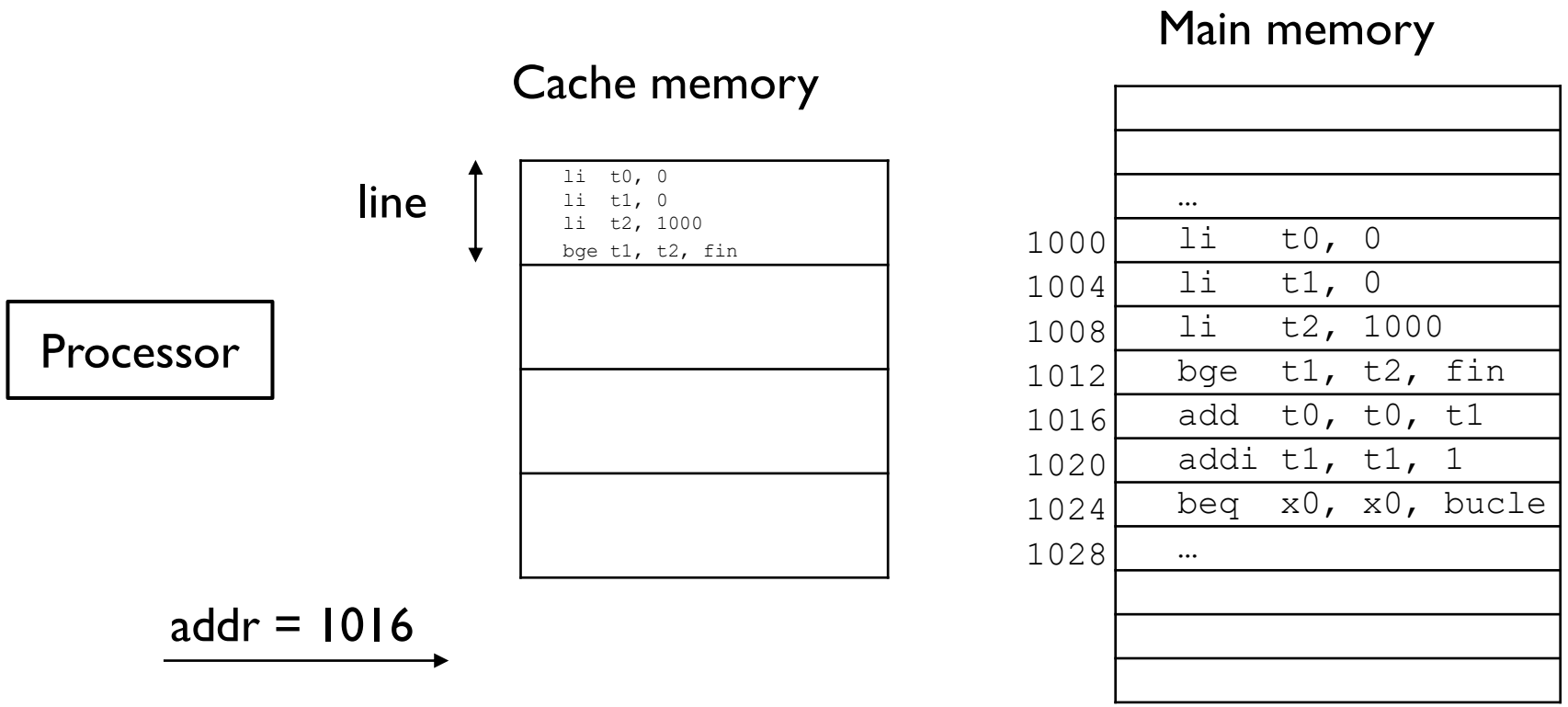
Example of how it works



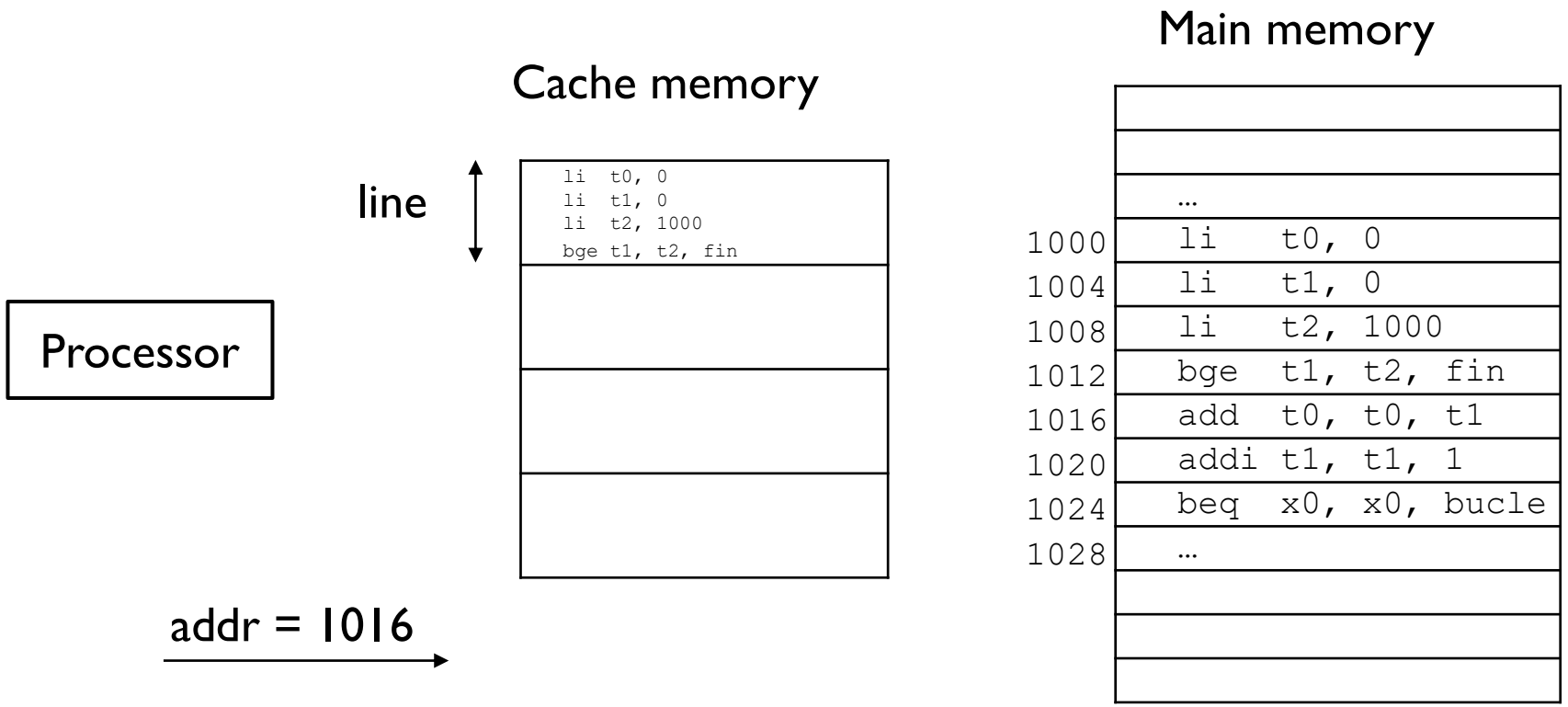
Example of how it works



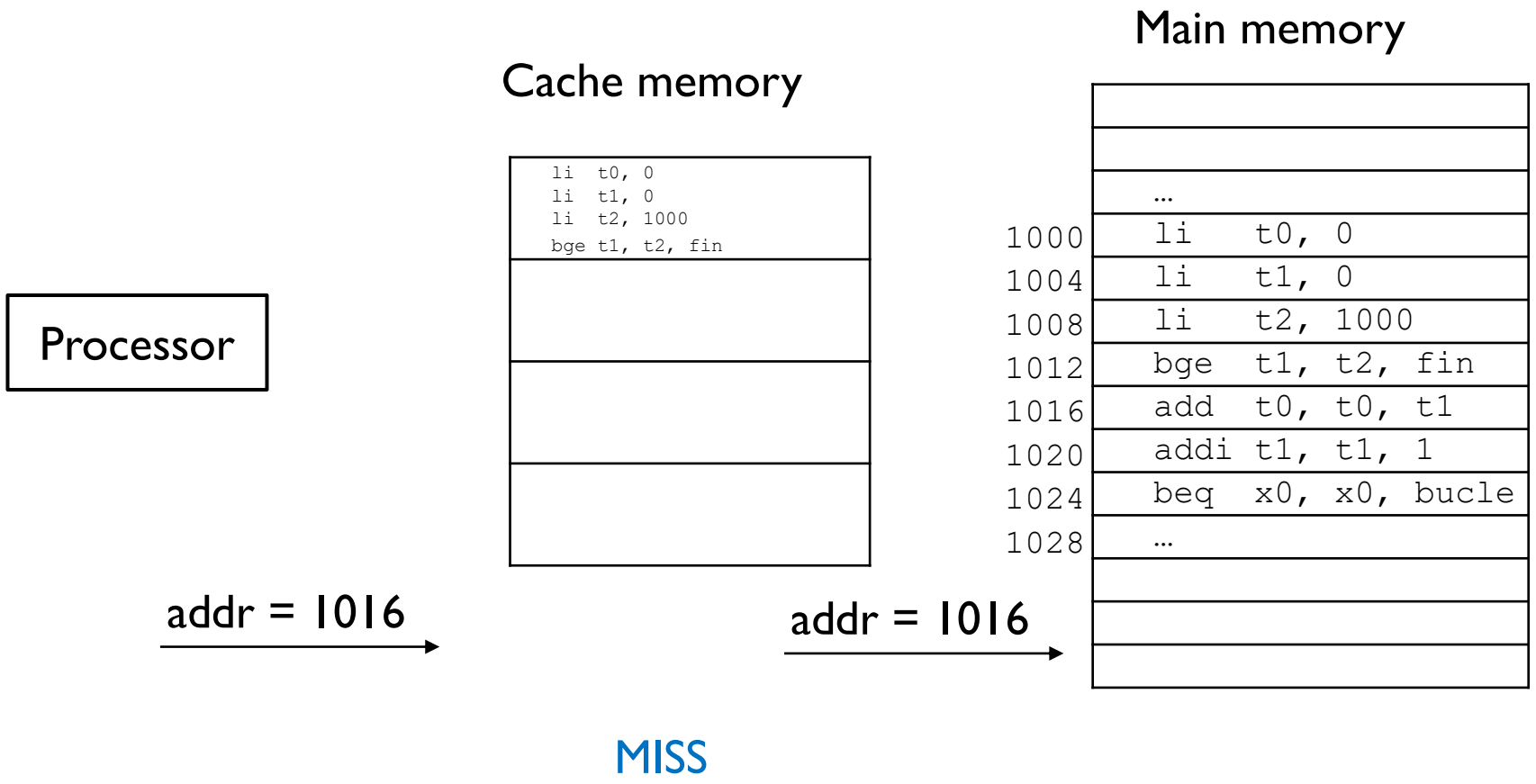
Example of how it works



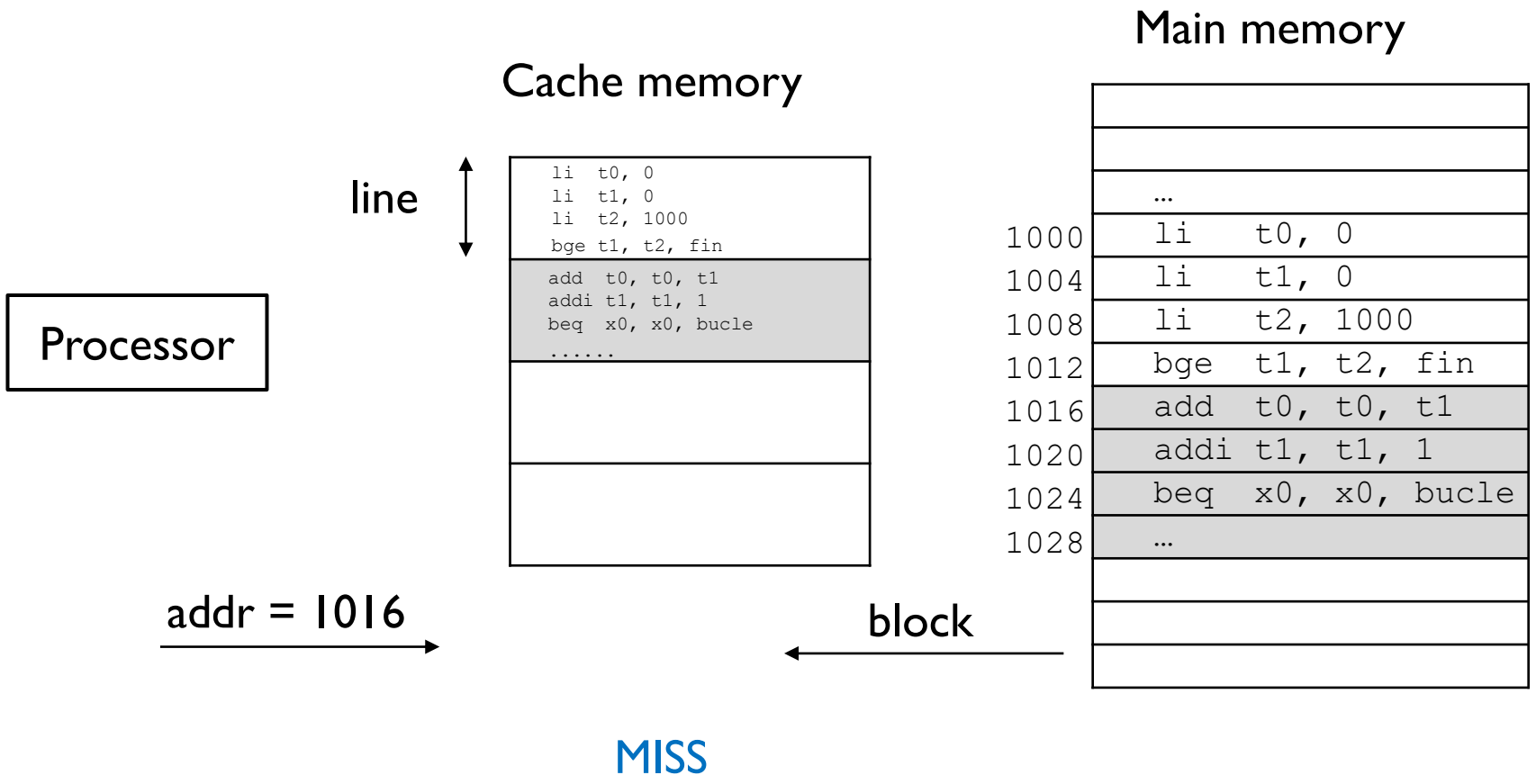
Example of how it works



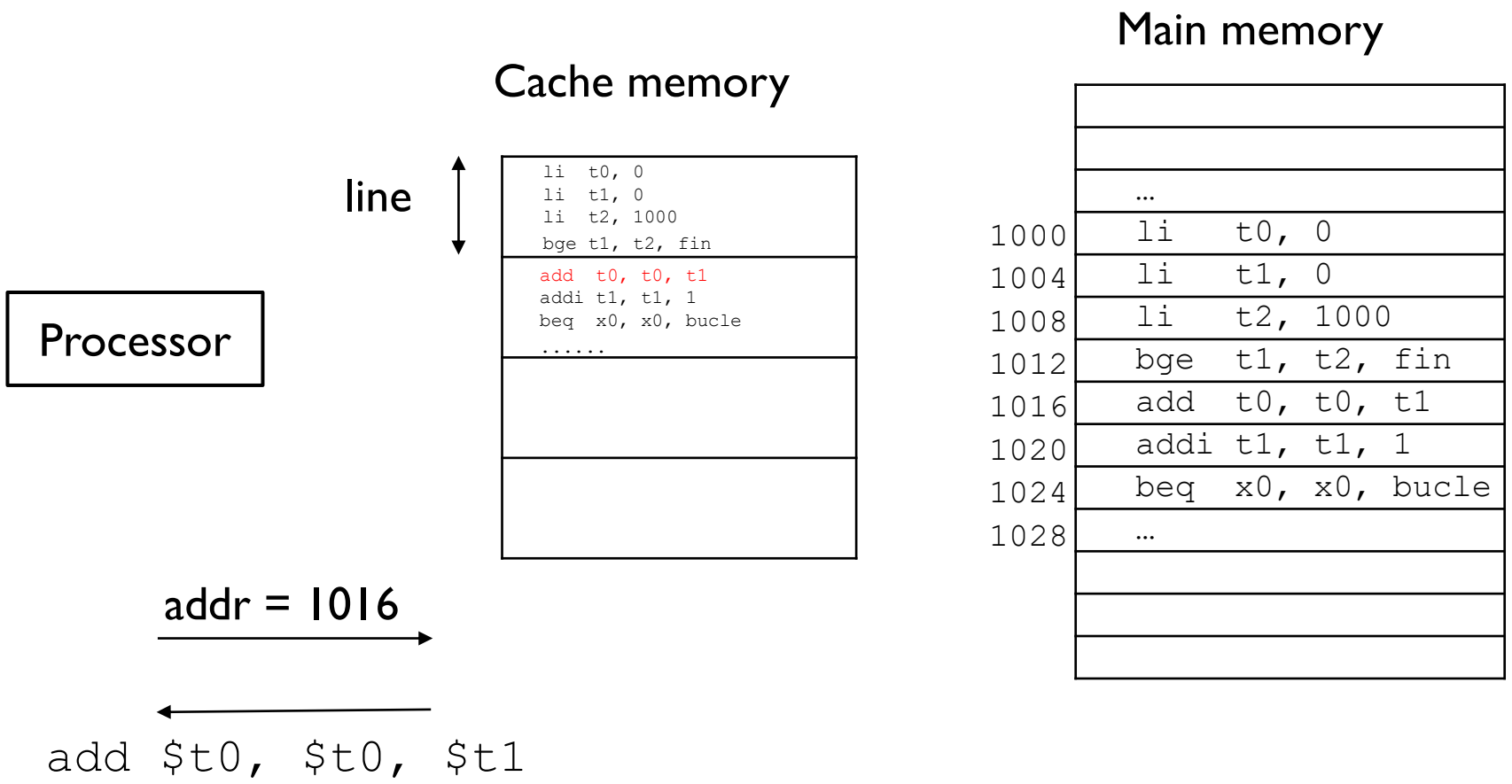
Example of how it works



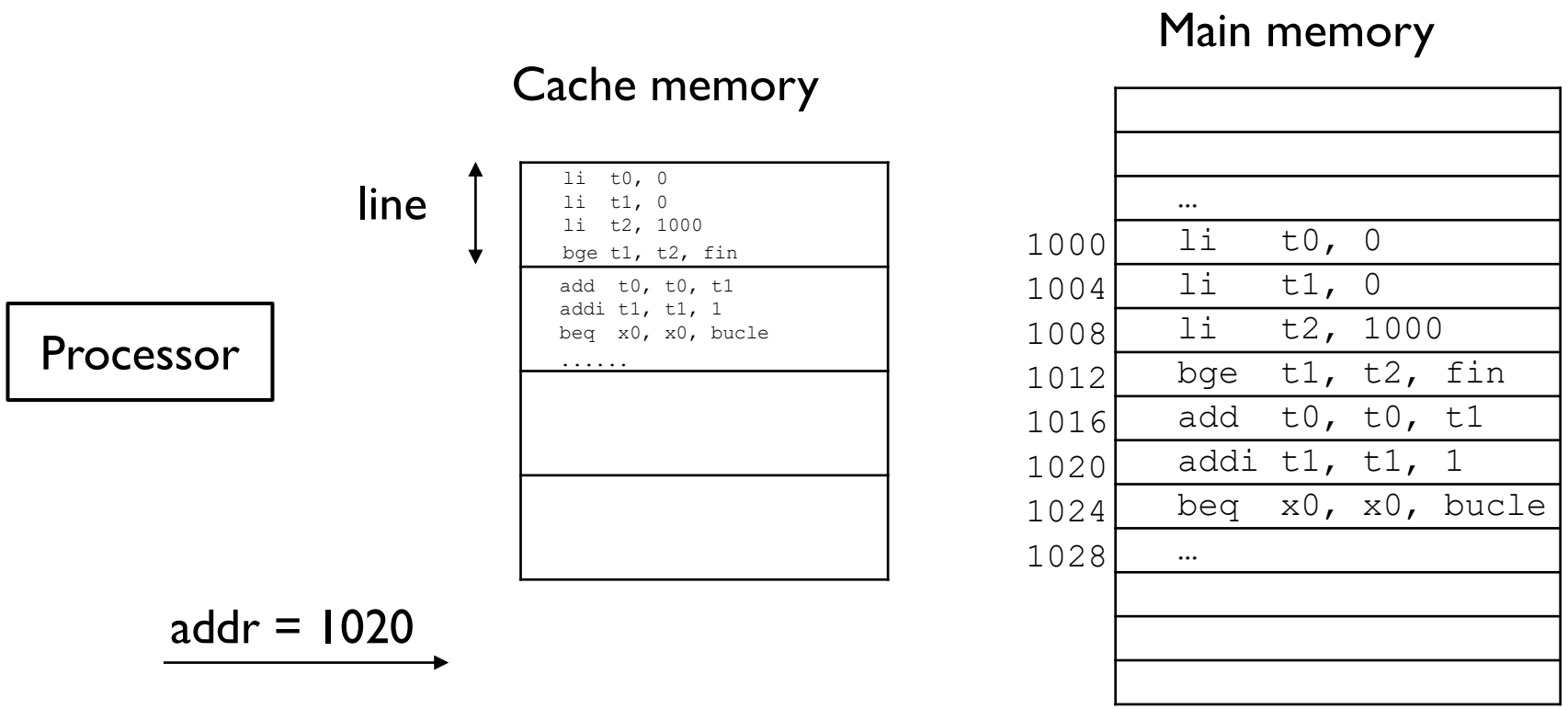
Example of how it works



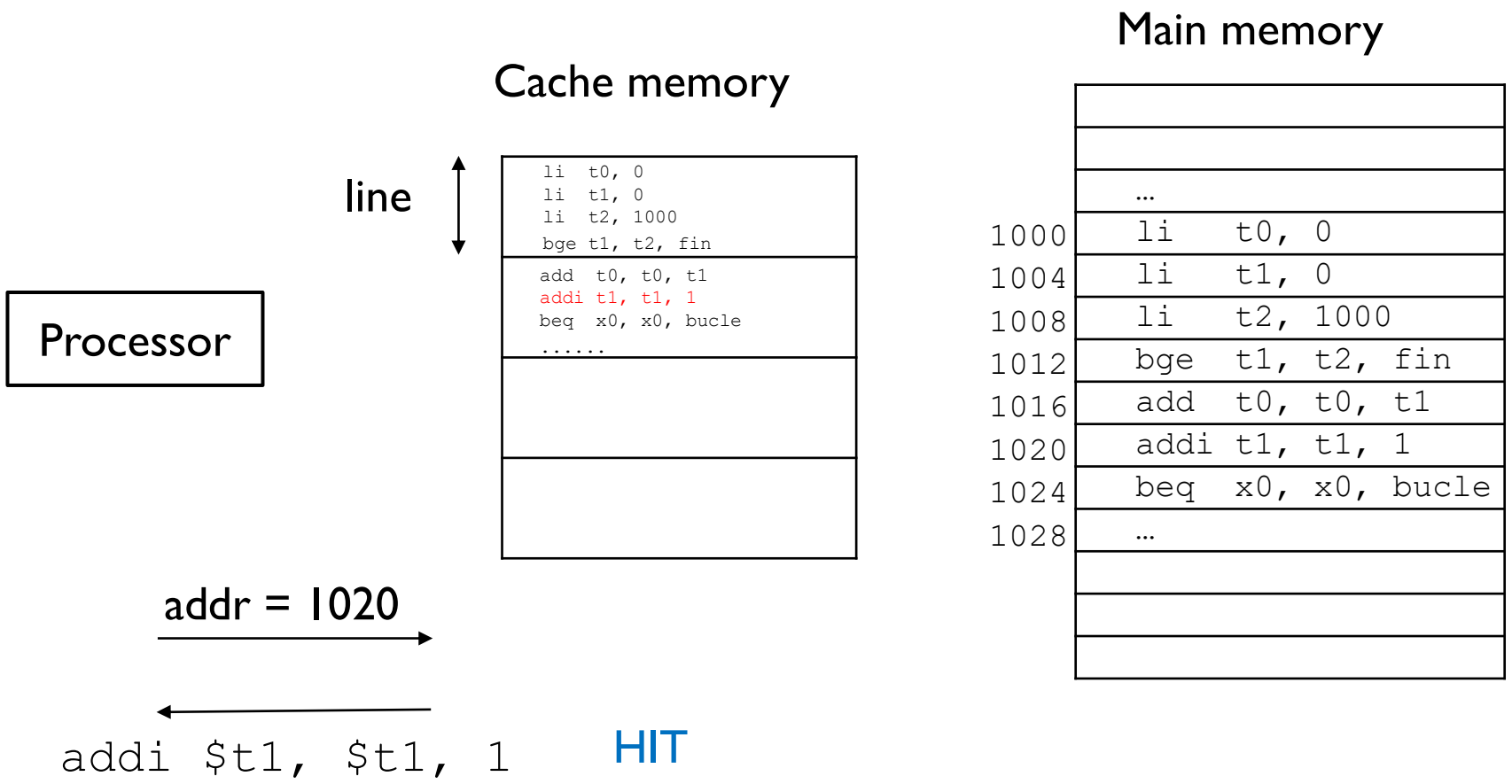
Example of how it works



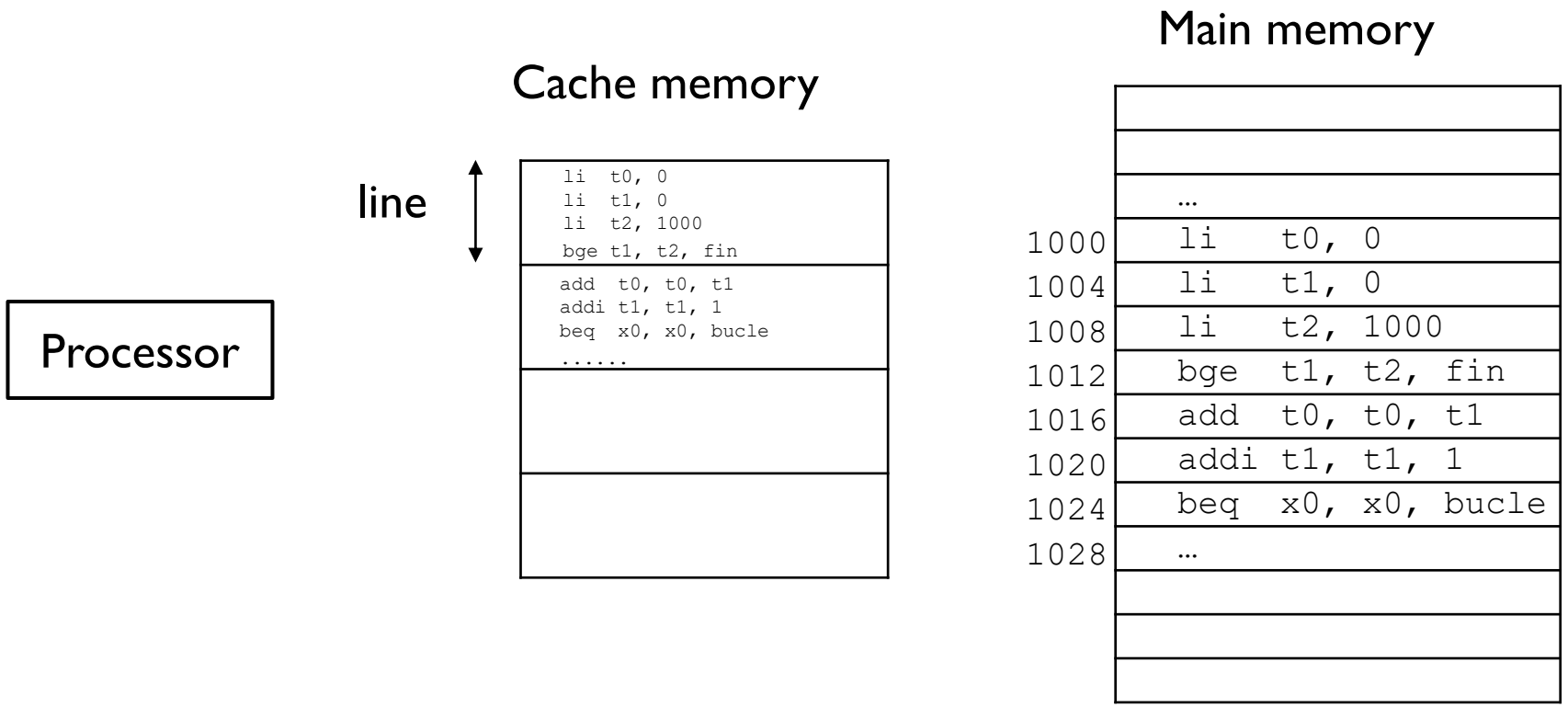
Example of how it works



Example of how it works



Example of how it works



All other accesses are HITS

Example of how it works

```
int i;
int s = 0;
for (i=0; i < 1000; i++)
     s = s + i;
```

```
        li   t0, 0     # s
        li   t1, 0     # i
        li   t2, 1000
loop1:  bge  t1, t2, end1
        add  t0, t0, t1
        addi t1, t1, 1
        beq  x0, x0, loop1
end1:    …
```

- With cache memory (blocks of 4 words):
  - Number of blocks = 2
  - Number of misses = 2

Example of how it works

```
int i;
int s = 0;
for (i=0; i < 1000; i++)
     s = s + i;
```

```
        li   t0, 0     # s
        li   t1, 0     # i
        li   t2, 1000
loop1:  bge  t1, t2, end1
        add  t0, t0, t1
        addi t1, t1, 1
        beq  x0, x0, loop1
end1:    …
```

- with cache memory:
  - Number of blocks= 2
  - Number of miss= 2
  - Time to transfer 2 blocks = 200 × 2 = 400 ns
  - Time to access the cache= 4004 × 2 = 8,008 ns
  - Total time= 8,408 ns

- **Cache access: 2 ns**
- **MP access: 120 ns**
- **MP block: 4 words**
- **Transfer a block between main memory and cache: 200 ns**

Example of how it works

```
int i;
int s = 0;
for (i=0; i < 1000; i++)
     s = s + i;
```

```
        li   t0, 0     # s
        li   t1, 0     # i
        li   t2, 1000
loop1: bge  t1, t2, end1
       add  t0, t0, t1
       addi t1, t1, 1
       beq  x0, x0, loop1
end1:   …
```

- Without cache memory = 480,480 ns
- With cache memory = 8,408 ns
- Hit ratio = 4002 / 4004 => 99,95 %

Exercise
Compute the hit ratio

```
int v[1000];  // global

int i;
int s;
for (i=0; i < 1000; i++)
     s = s + v[i];
```

- Exercise:
  - Cache access: 2 ns
  - MP access: 120 ns
  - MP block: 4 words
  - Transfer a block between main memory and cache: 200 ns

```
.data
        v: .space 4000 # 4*1000

.text
main:
        li   t0, 0    # i
        li   t1, 0    # i de v
        li   t2, 1000 # componentes
        li   t3, 0    # s
bucle: bge  t0, t2, fin
        lw   t4, v(t1)
        add  t3, t3, t4
        addi t0, t0, 1
        addi t1, t1, 4
        beq  x0, x0, bucle
fin:    …
```

Why does the cache work?

- Cache access time is much shorter than main memory access time.
- Main memory is accessed by blocks.
- When a program accesses an address, it is likely to access it again in the near future.
  - Temporary locality.
- When a program accesses an address, it is likely to access nearby positions in the near future.
  - Spatial location.
- Hit ratio: probability that an accessed data is in cache

High hit ratio

Typical access time

- Main memory.
  - DRAM technology or similar.
  - Access time: 20 - 50 ns.

- Cache memory.
  - SRAM technology or similar.
  - Access time: < 1 – 2.5 ns.

Average cache access time

- Average access time of a two-level memory system

$$Tm = h \cdot Ta + (1-h) \cdot (Ta+Tf)$$
$$= Ta + (1-h) \cdot Tf$$

- Ta: cache access time
- Tf: time to process a miss, time to replace a block and bring it from main memory to cache
- h: hit ratio

Overall performance

$$Tm = h \cdot Ta + (1-h) \cdot (Ta+Tf)$$
$$= Ta + (1-h) \cdot Tf$$

1. The Processor performs an access to cache.
2. The cache checks if the data for this position is already there:

- **If it is there (HIT),**

  3.A.1 It is served to the Processor from the cache (quickly): Ta

- **If it is not there (MISS),**

  3.B.1 The cache transfers from Main memory the block associated with position: Tf

  3.B.2 The cache then delivers the requested data to the processor: Ta

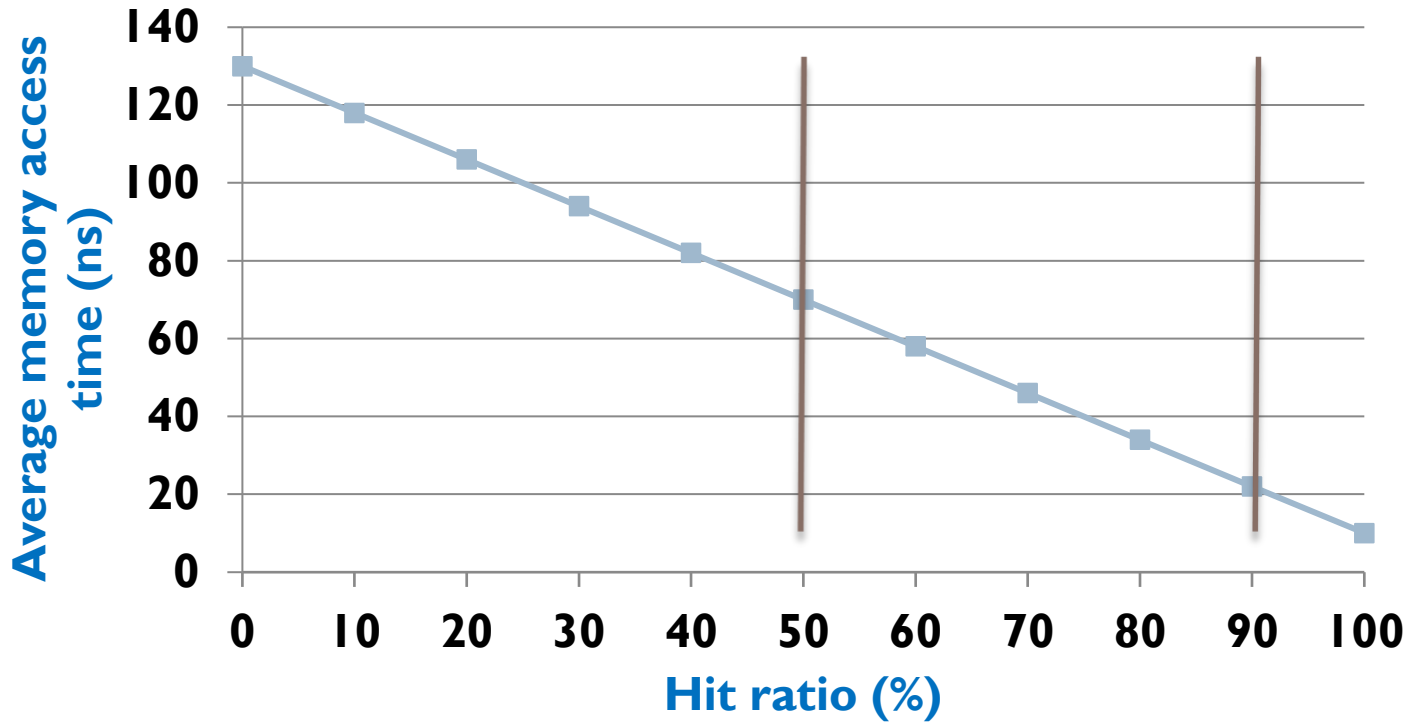
Example

$$Tm = h \cdot Ta + (1-h) \cdot (Ta+Tf)$$
$$= Ta + (1-h) \cdot Tf$$

1. Ta: Cache access time = **10** ns
2. Tf: Main memory access time = **120** ns
3. h: Hit ratio-> **X = 0.1, 0.2, …, 0.9, 1.0**
   10%, 20%, …, 90%, 100%

Example

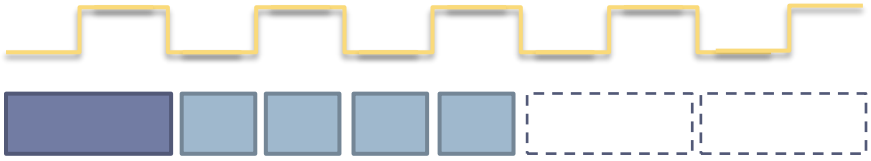
$$Tm = h \cdot Ta + (1-h) \cdot (Ta+Tf)$$
$$= Ta + (1-h) \cdot Tf$$



Block access
review

- It is better to access to consecutive words
  - Example 1: access to 5 individuals non-consecutives words

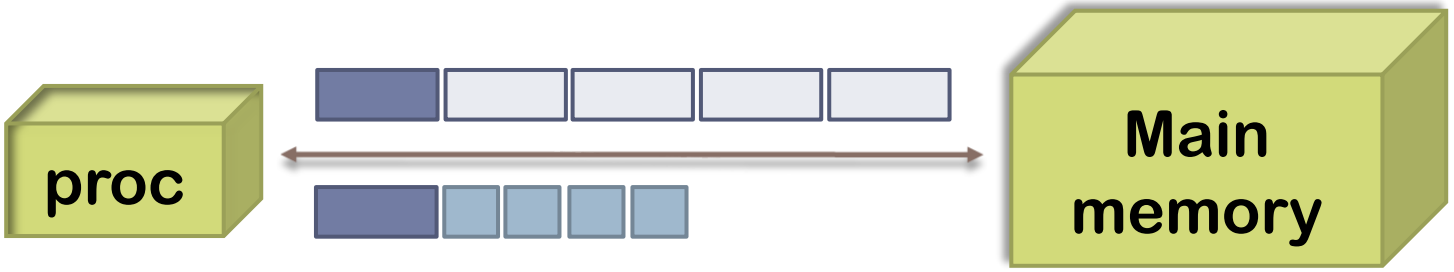
  - Example 2: access to 5 consecutives words



Block access

review

- If a consecutive set of memory locations is accessed, subsequent accesses to the first one have a lower cost.



Cache memory levels

- It is common to find three levels:
  - L1 or level 1:
    - Internal cache: closest to the Processor
    - Small size (8KB-128KB) and maximum speed
    - Can be split for instructions and data
  - L2 or level 2:
    - Internal cache
    - Between L1 and L3 (or between L1 and main memory)
    - Medium size (256KB - 4MB) and lower speed than L1
  - L3 or level 3:
    - Typically, last level before main memory
    - Larger size and slower speed than L2
    - Internal or external to the processor

Exercise

- Computer:
  - Cache access time: 4 ns
  - Main memory block access time: 120 ns.
- If the Hit ratio is 90%, what is the average access time?
- Hit rate required for a mean access time to be less than 5 ns.

Exercise

- Computer:
  - Cache access time: 4 ns
  - Time to access a block of Main Memory: 120 ns.
- With a hit ratio of 90%, what is the average memory access time?
- What is the hit ratio needed to obtain a memory access time less than 10 ns?

Exercise (solution)

- Computer:
  - Cache access time: 4 ns
  - Time to access a block of Main Memory: 120 ns.
- With a hit ratio of 90%, what is the average memory access time?

$$T_m = 4 \times 0.9 + (120 + 4) \times 0.1 = 16\, ns$$

- What is the hit ratio needed to obtain a memory access time less than 10 ns?

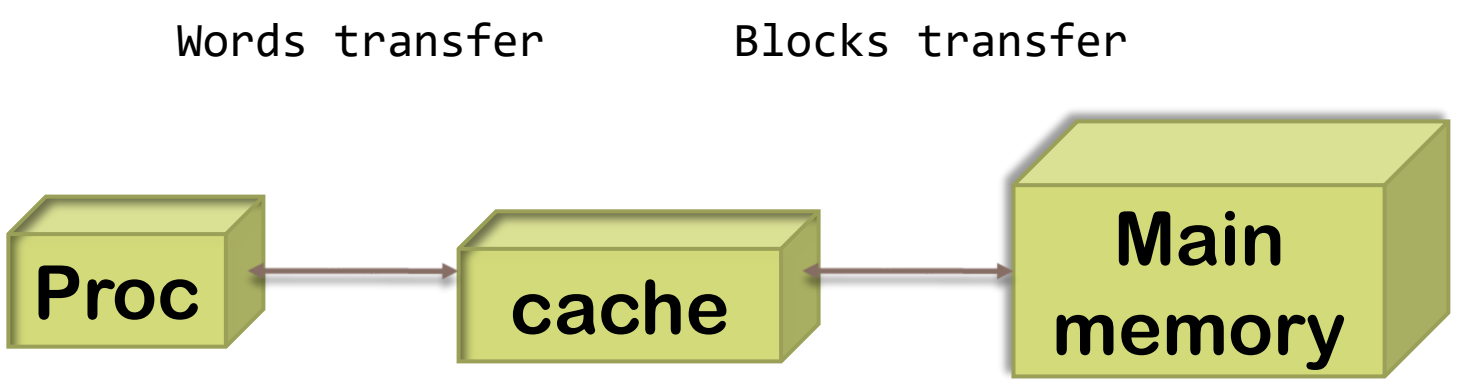
$$5 = 4 \times h + (120 + 4) \times (1 - h) \qquad \Rightarrow$$

$$\Rightarrow h > 0.9916$$

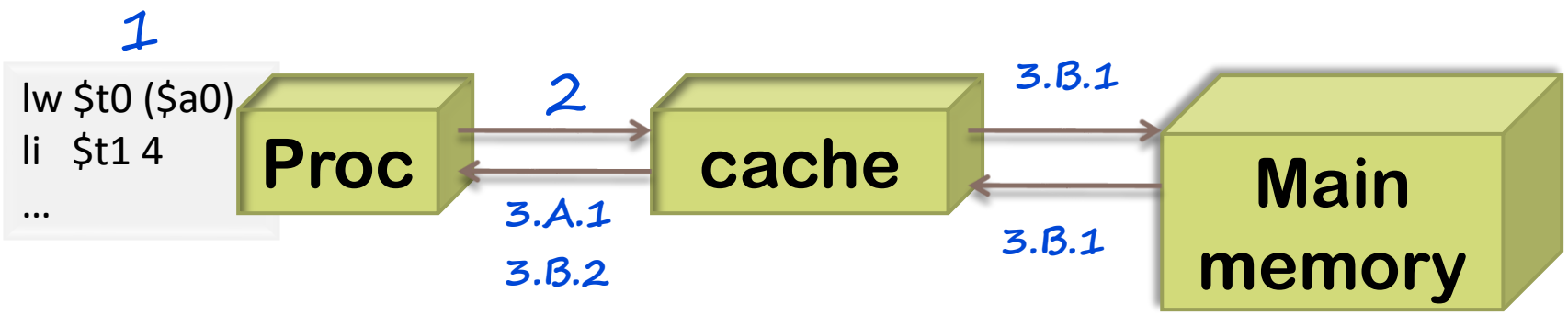
Cache memory
summary

- It is built with SRAM technology
  - Integrated in the Processor itself.
  - Faster and more expensive than DRAM memory.

- Maintains a copy of chunks of the main memory.

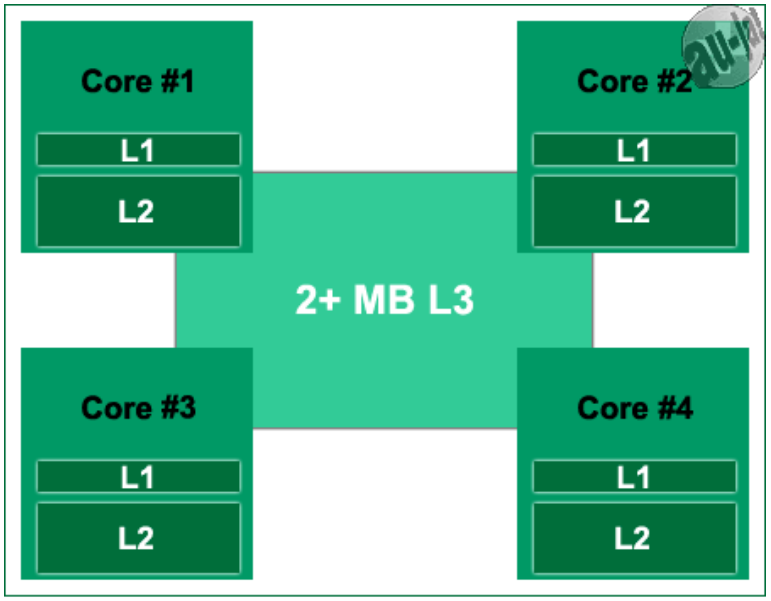


How it works (in general)
summary

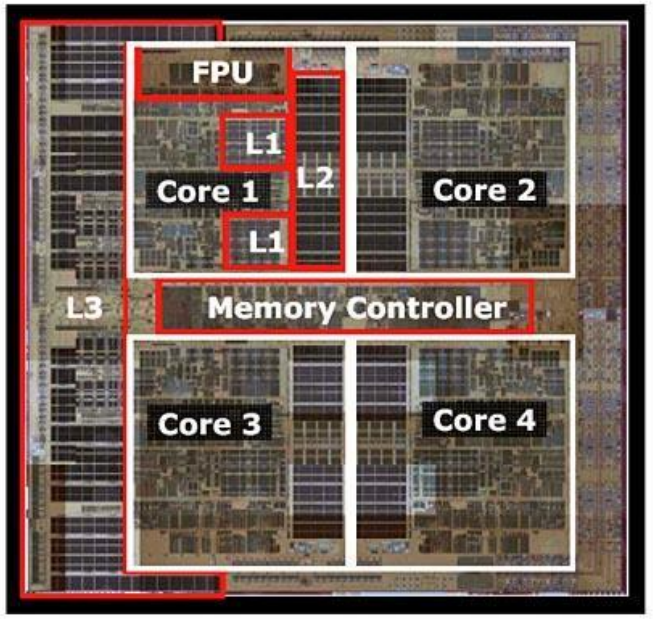


1. The Processor performs an access to cache.
2. The cache checks if the data for this position is already there:
   - **If it is there (HIT)**,
     - **3.A.1** It is served to the Processor from the cache (quickly): Ta
   - **If it is not there (MISS)**,
     - **3.B.1** The cache transfers from Main memory the block associated with position: Tf
     - **3.B.2** The cache then delivers the requested data to the processor: Ta

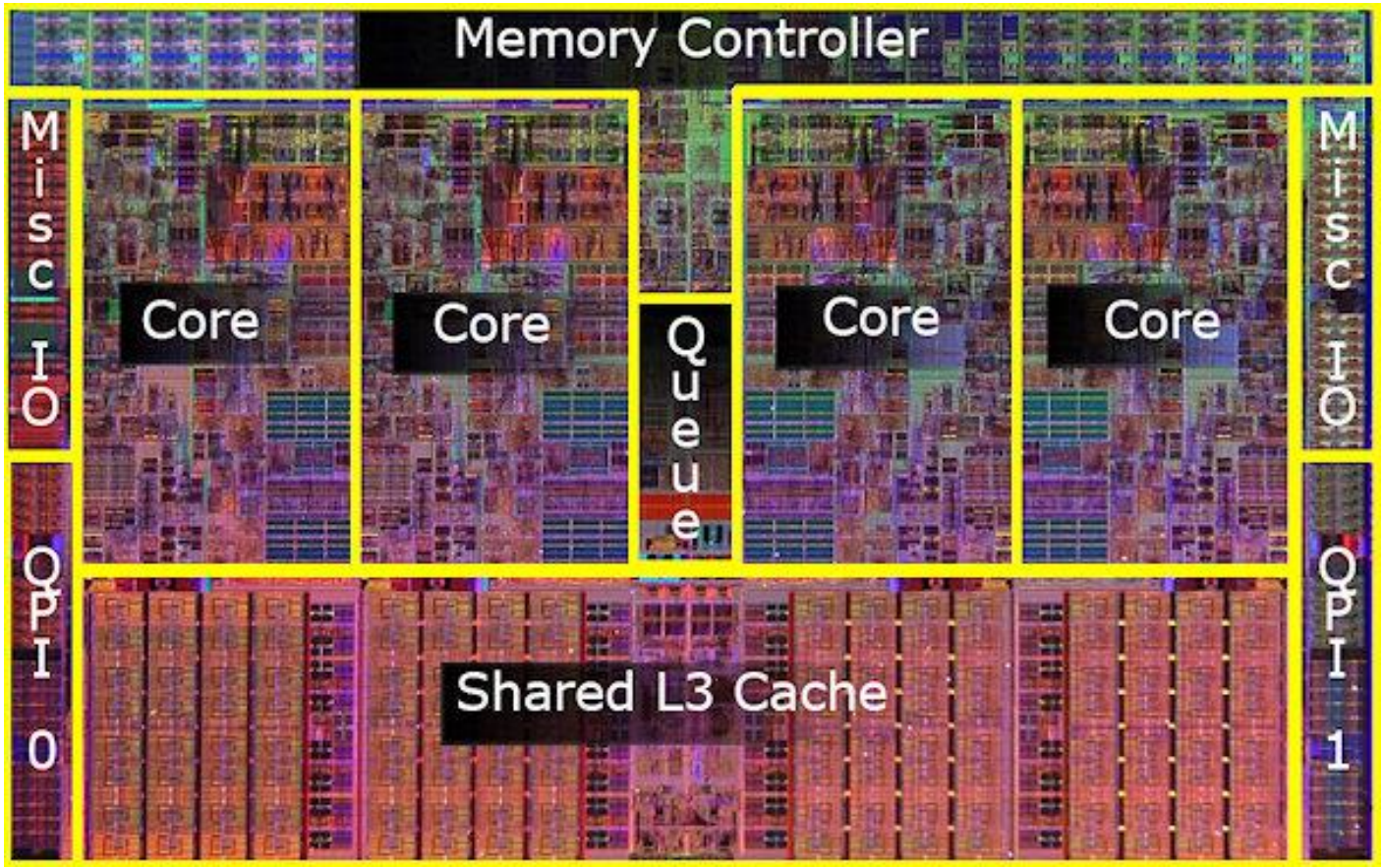
Example: AMD Quad-core



Quad-Core CPU mit gemeinsamen L3-Cache



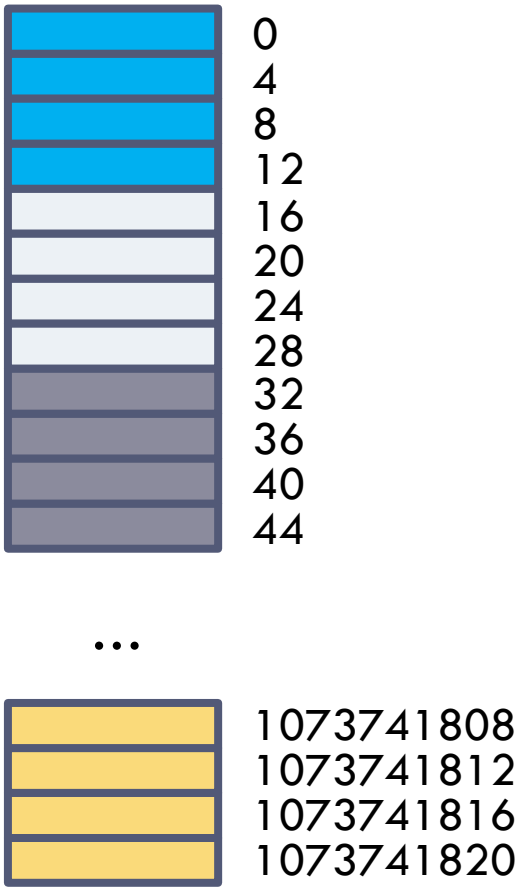
Example: Intel Core i7



Cache memory design and organization

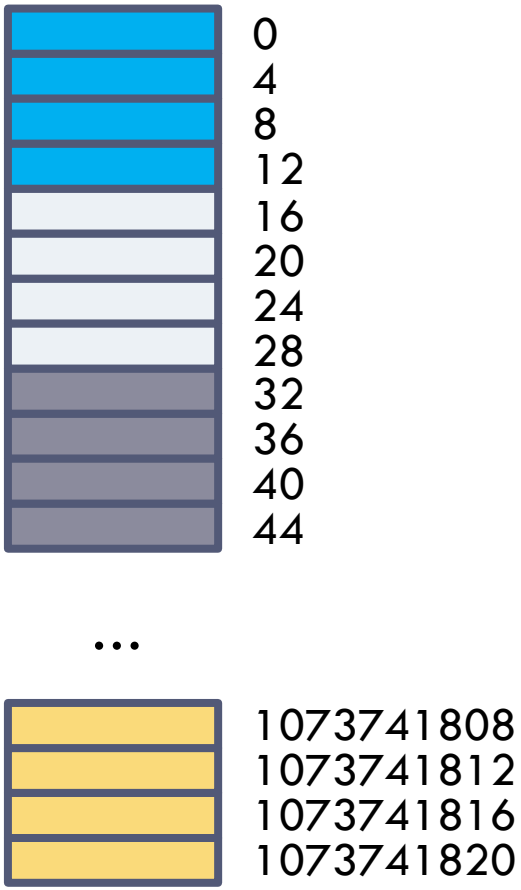
1. **Cache memory structure**
2. **Cache memory design**
   - Size
   - Matching function
   - Substitution algorithm
   - Write policy

Main memory organization



- Main memory *logically* divided in blocks of same size:
  - 1 block = k words
- Exercise: How many blocks (of 4 bytes) are there in a memory of 1 GB?

Main memory organization



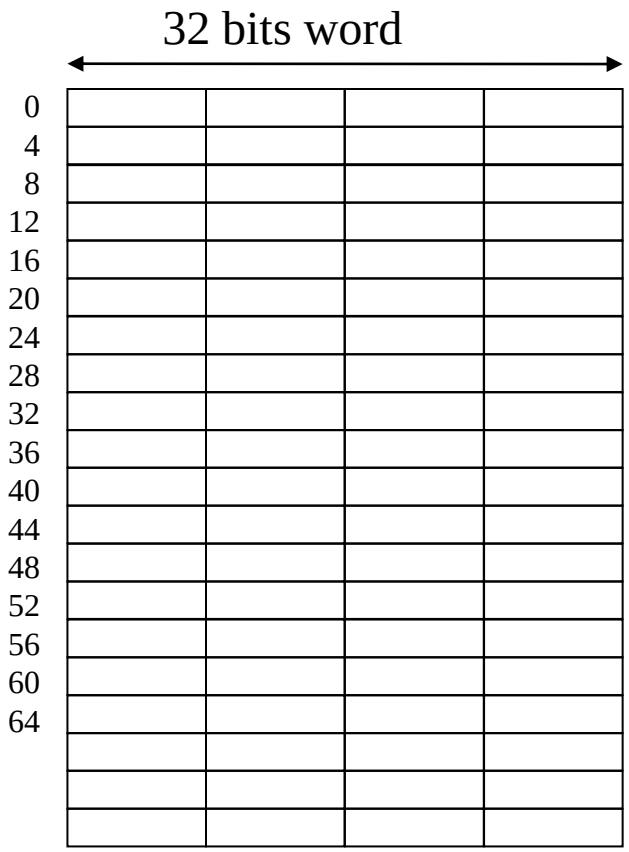
- Main memory *logically* divided in blocks of same size:
  - 1 block = k words
- Exercise: How many blocks (of 4 bytes) are there in a memory of 1 GB?
  - *solution*:

    2^{30} B / 16 B = 2^{30-4} B = 2^{26} B = 64 megablocks ≈ 64 millions

Main memory access

- Example:
  - 32 bit computer
  - Byte addressed memory
  - Main memory accessed by word
  - How to access to this address?

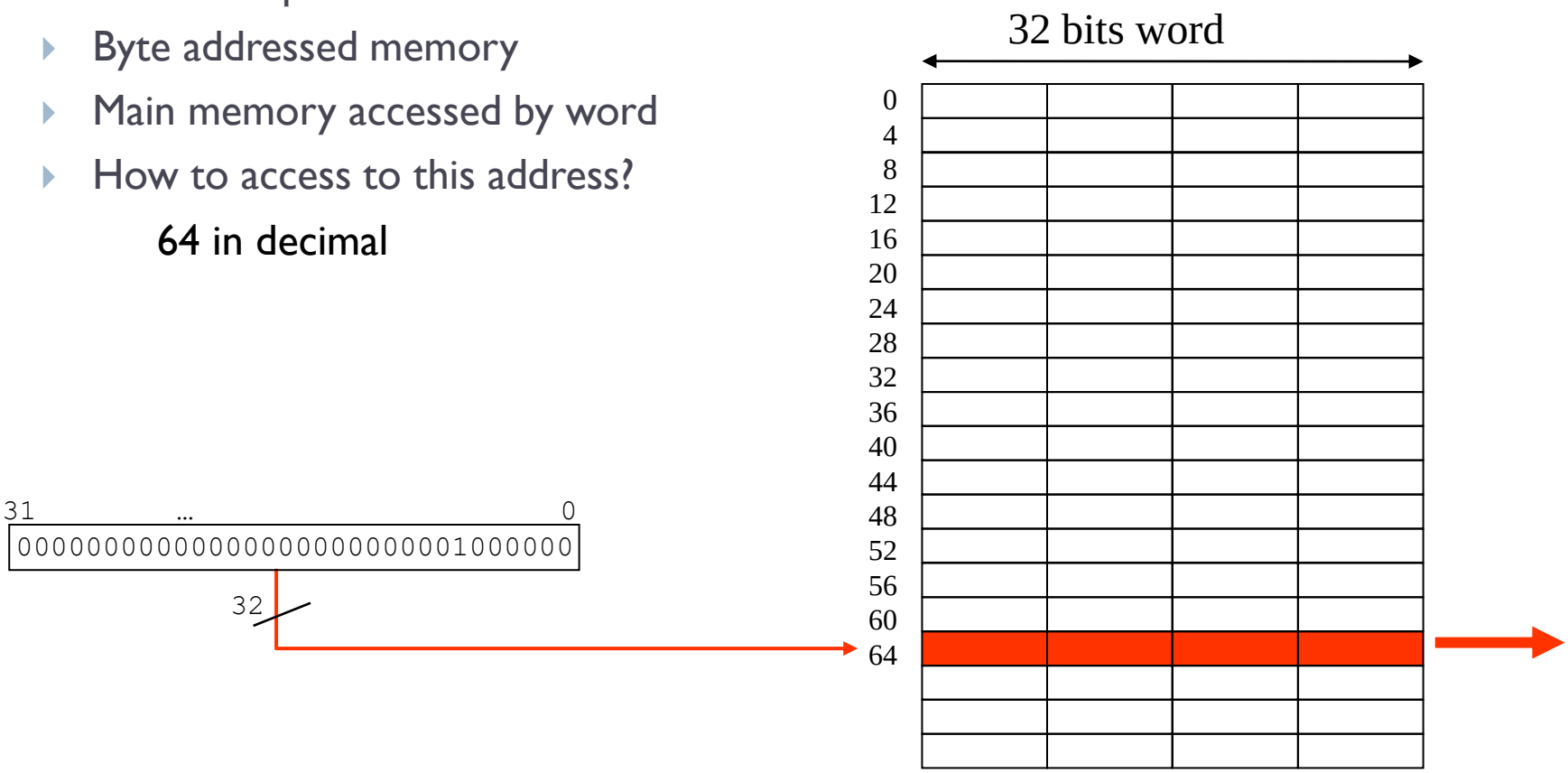
    64 in decimal



Main memory access

- Example:
  - 32 bit computer
  - Byte addressed memory
  - Main memory accessed by word
  - How to access to this address?

    64 in decimal



Cache memory organization

- Cache memory is organized in blocks (lines)
  - The Cache memory block is called cache line
- The size of a block from the main memory is equal to the size of the line.
  - But the size of the cache memory is much smaller.
  - The number of blocks that fit in the cache memory is very small.
- How many blocks of 4 words can fit in a 32 KB cache memory?

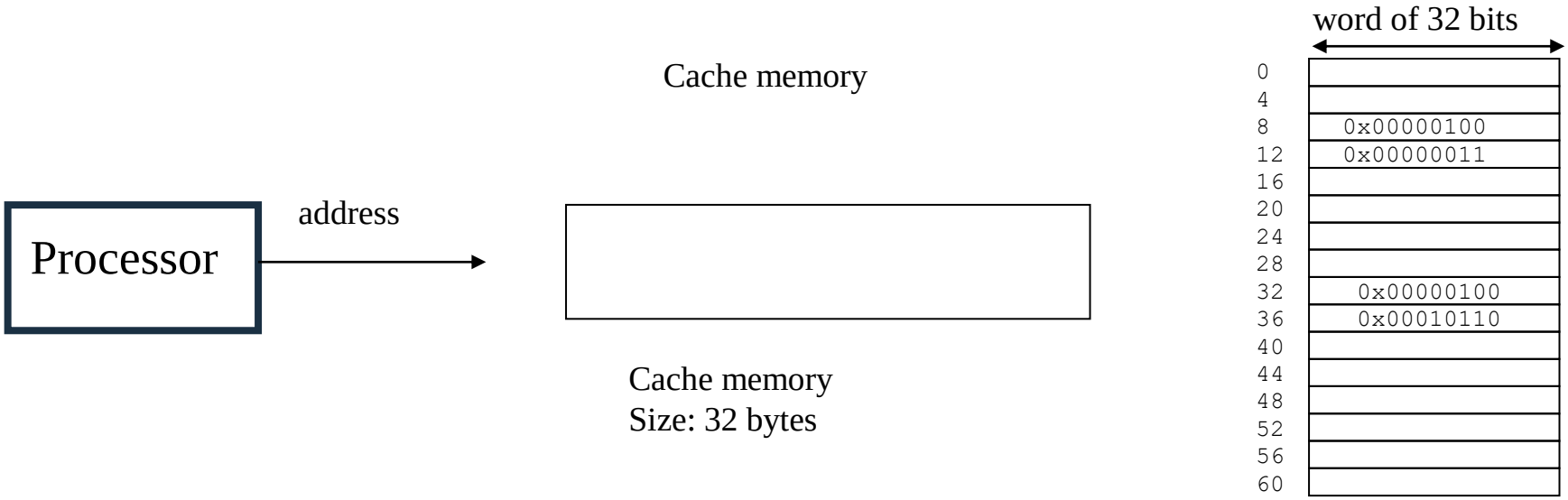
Cache memory organization

- Cache memory is organized in blocks (lines)
  - The Cache memory block is called cache line
- The size of a block from the main memory is equal to the size of the line.
  - But the size of the cache memory is much smaller.
  - The number of blocks that fit in the cache memory is very small.
- How many blocks of 4 words can fit in a 32 KB cache memory?
  - solution:

    $2^5 \cdot 2^{10}$ B / 2^4 B = 2^{11} blocks = 2048 blocks = 2048 lines

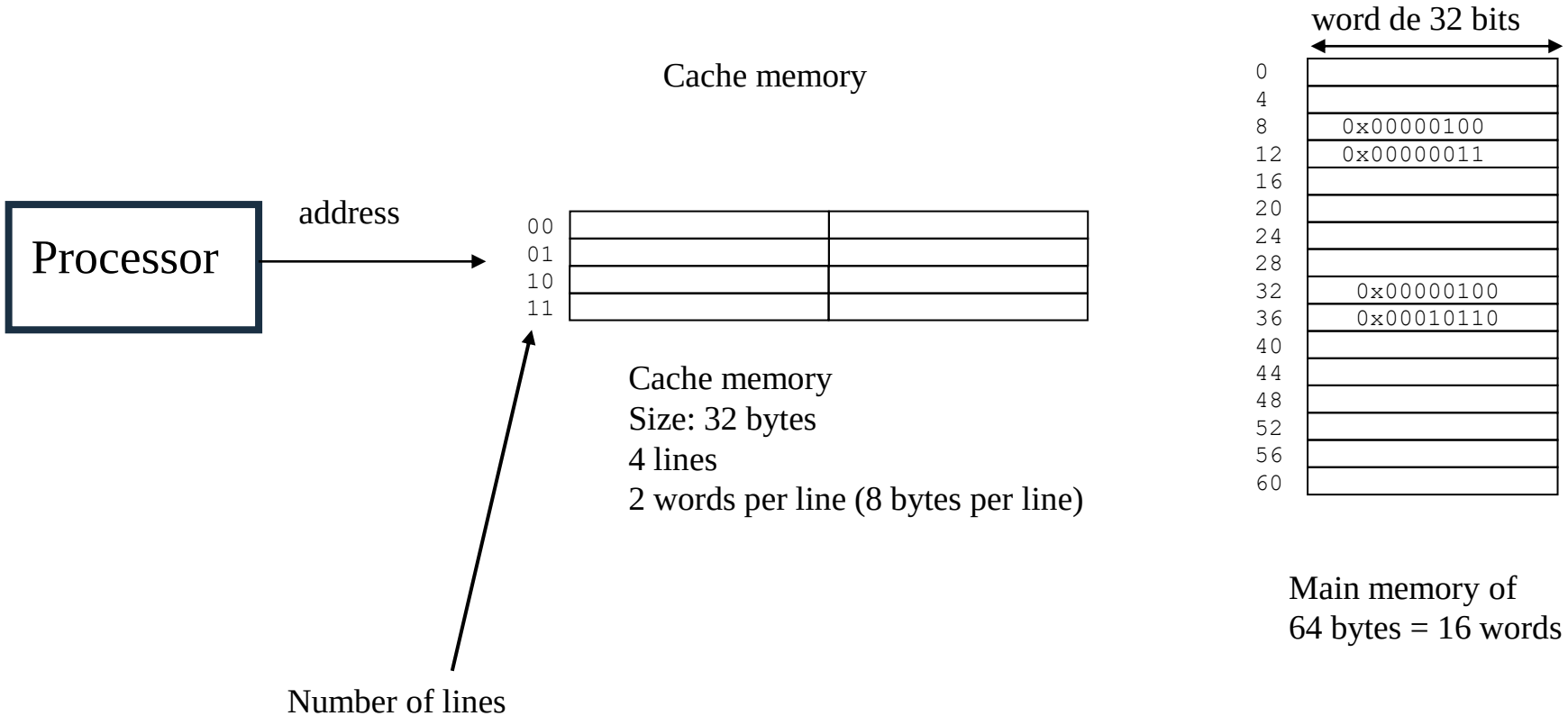
Locating a word in the cache

Example:

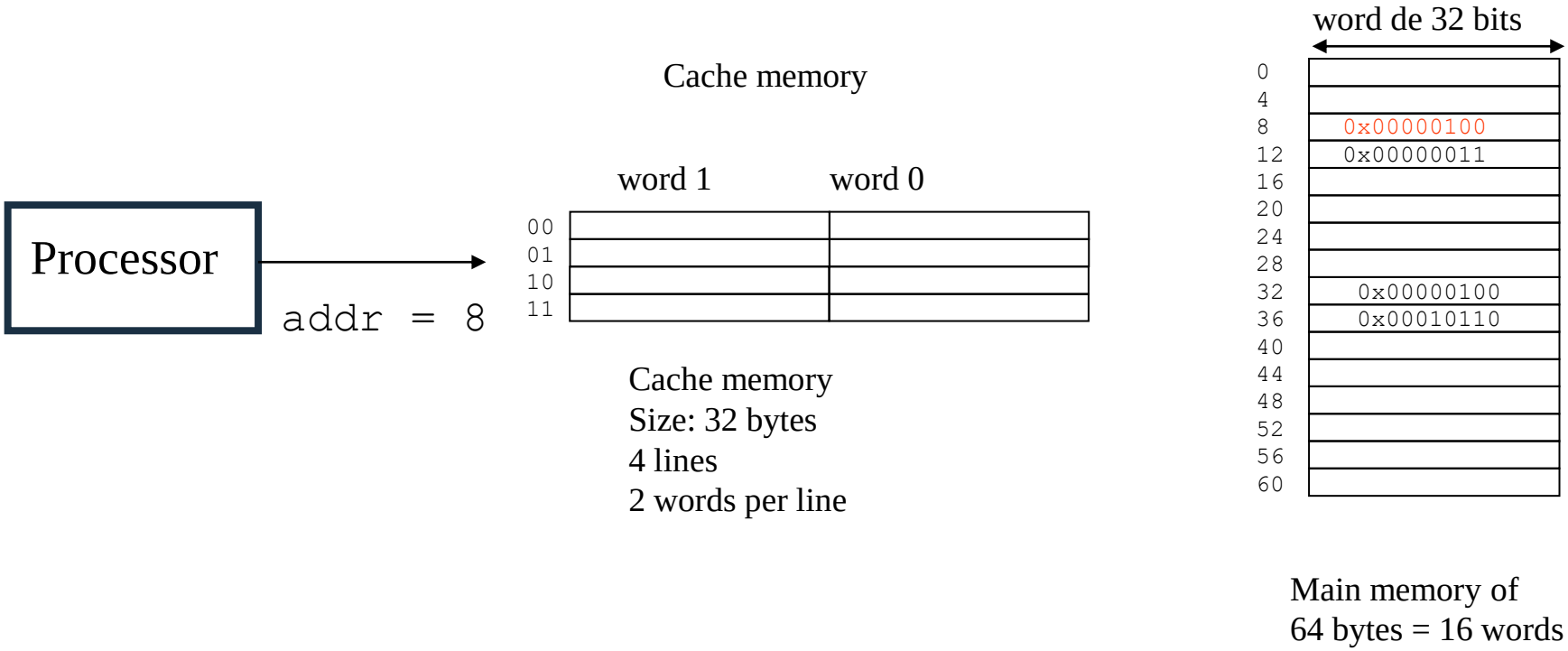


With blocks of 2 words
How many lines does the cache have?

Locating a word in the cache



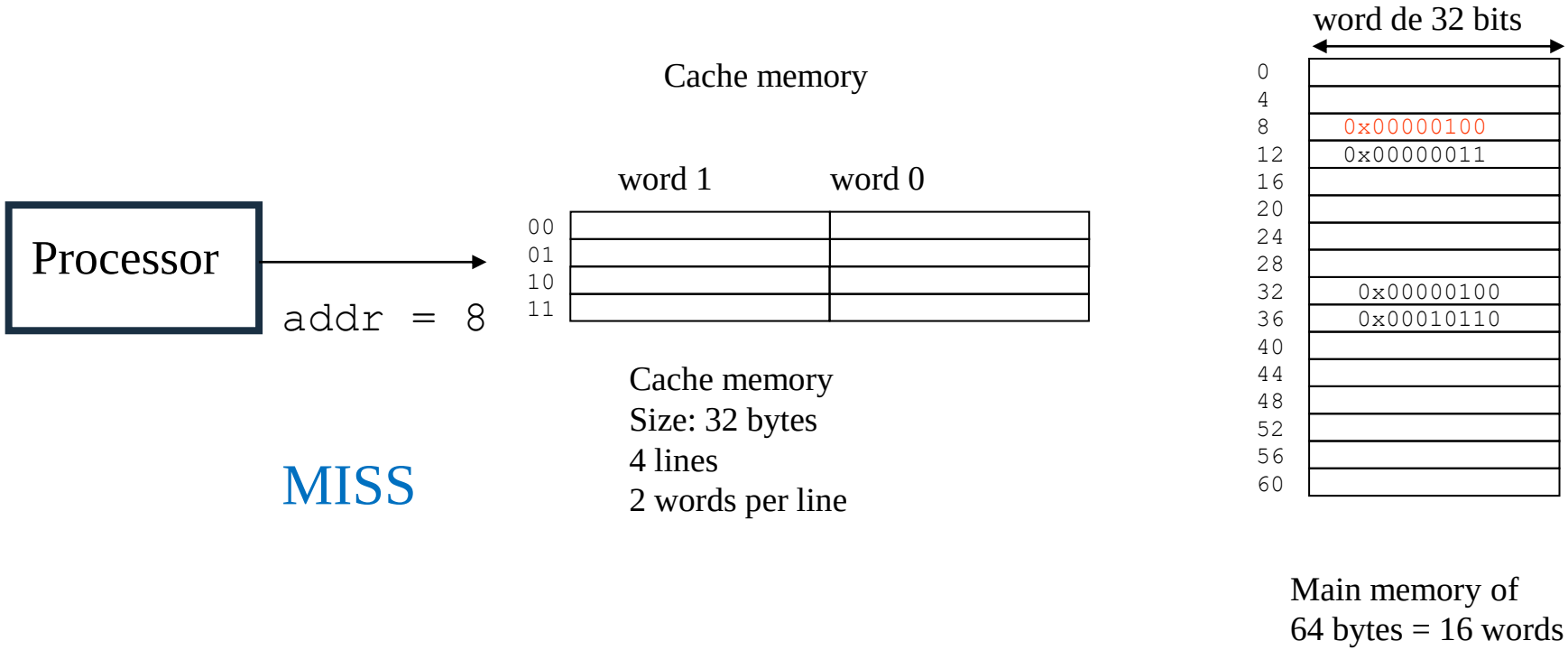
Locating a word in the cache



Cache memory
Size: 32 bytes
4 lines
2 words per line

Main memory of
64 bytes = 16 words

Locating a word in the cache



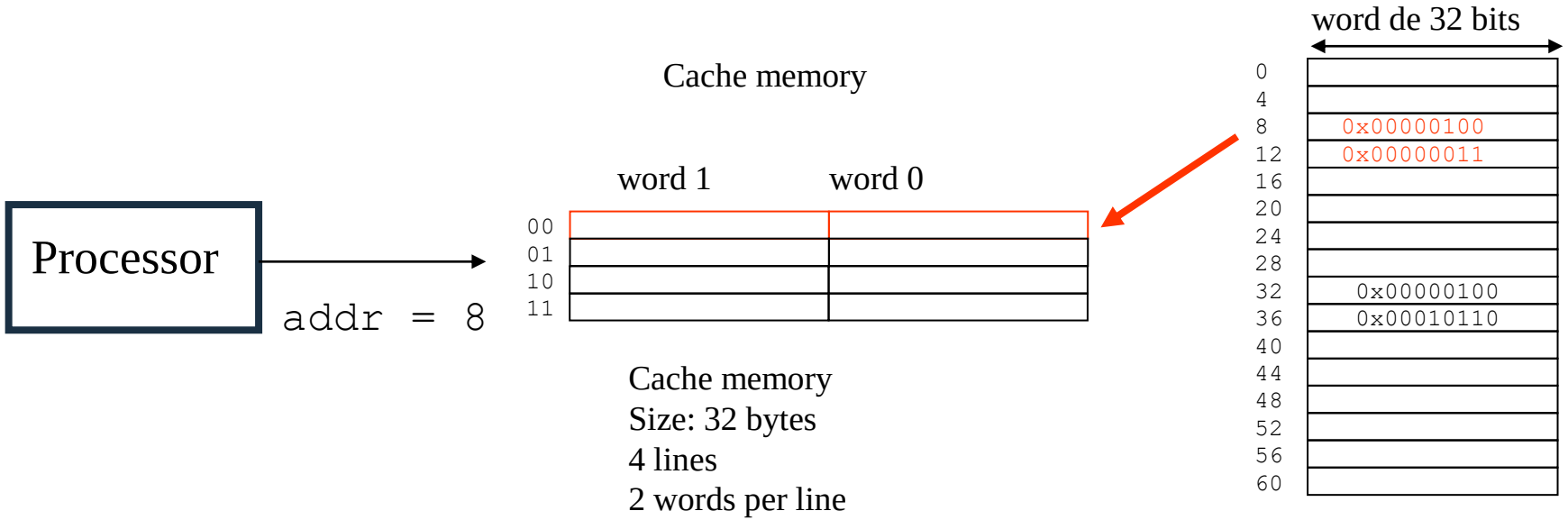
Cache memory
Size: 32 bytes
4 lines
2 words per line

Main memory of
64 bytes = 16 words

MISS

How is it known?

Locating a word in the cache

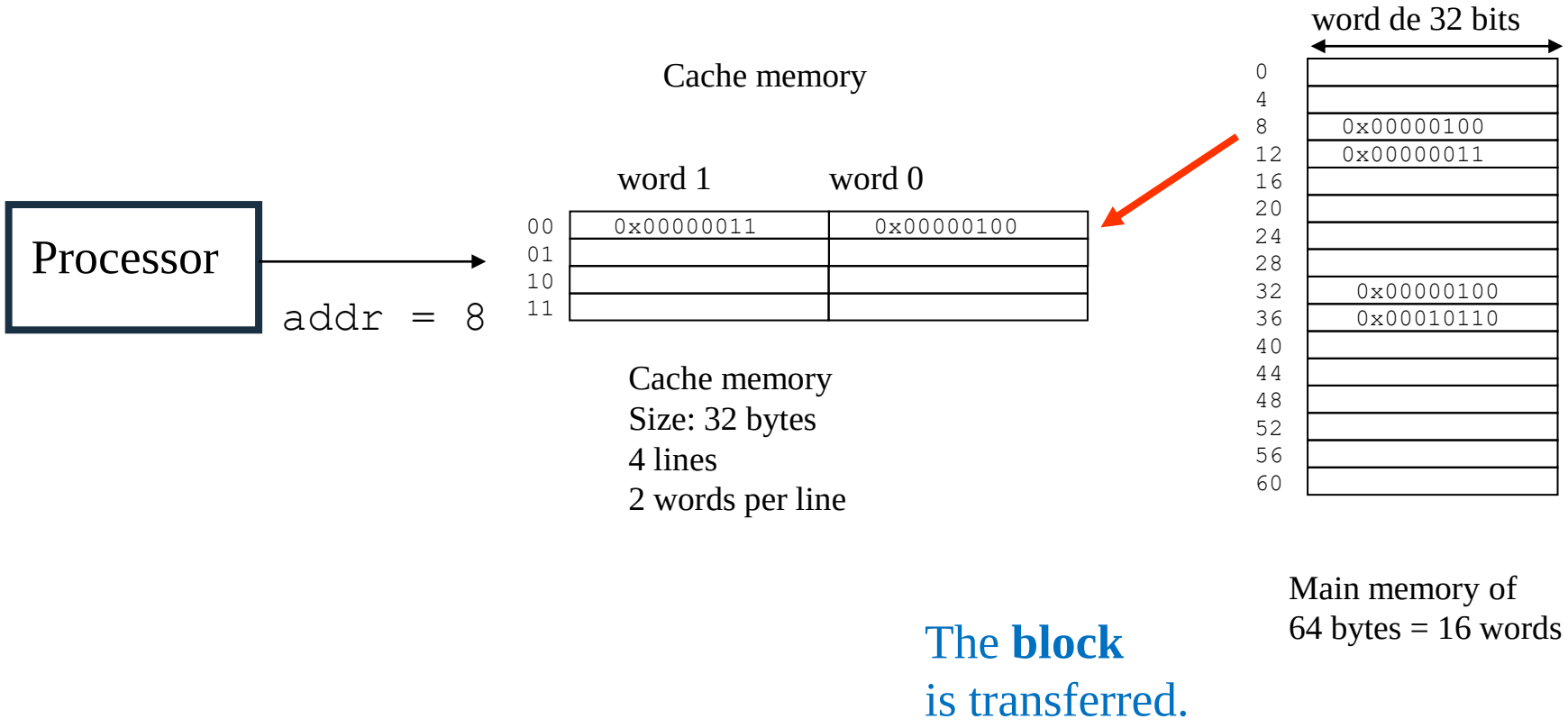


Cache memory
Size: 32 bytes
4 lines
2 words per line

Main memory of
64 bytes = 16 words

A line is selected in the cache.
Which line?

Locating a word in the cache

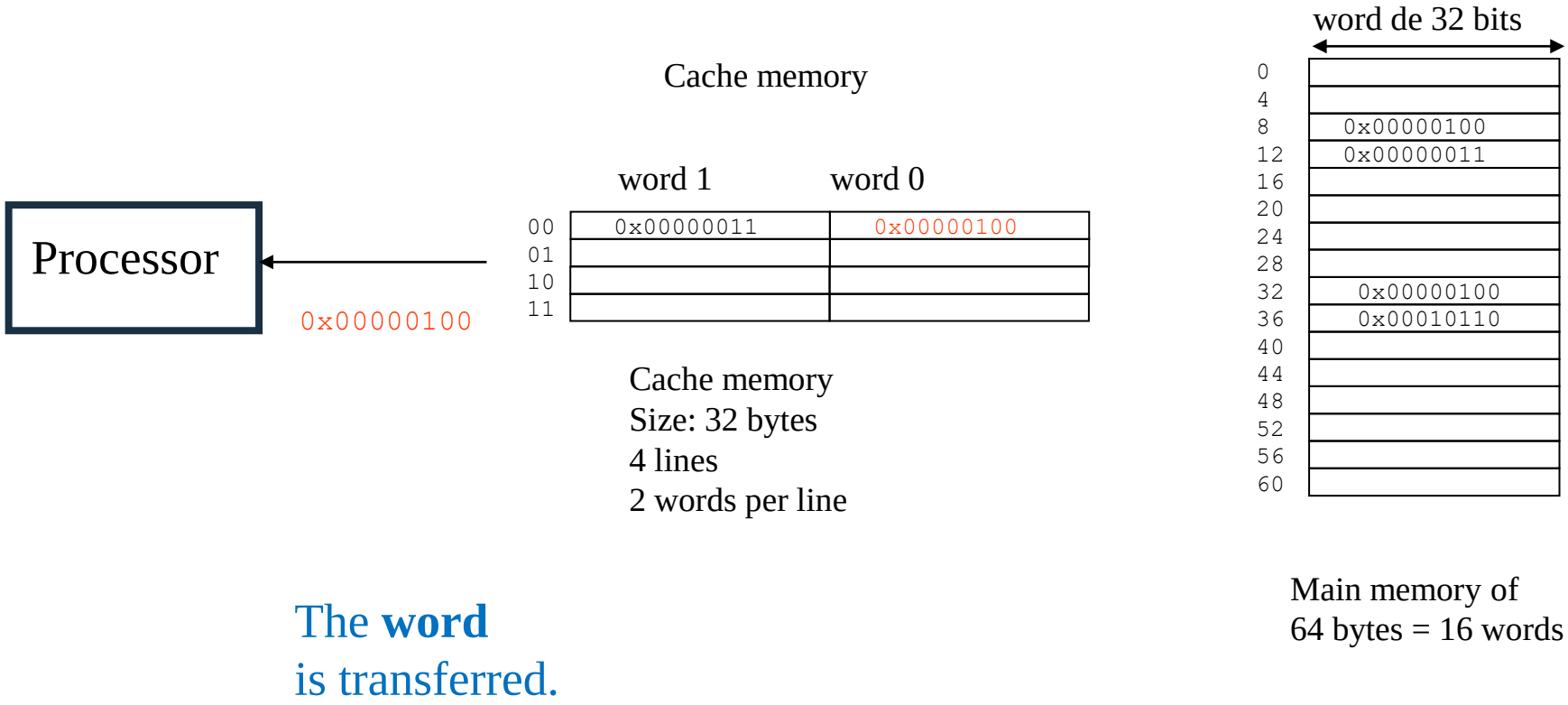


Cache memory
Size: 32 bytes
4 lines
2 words per line

Main memory of
64 bytes = 16 words

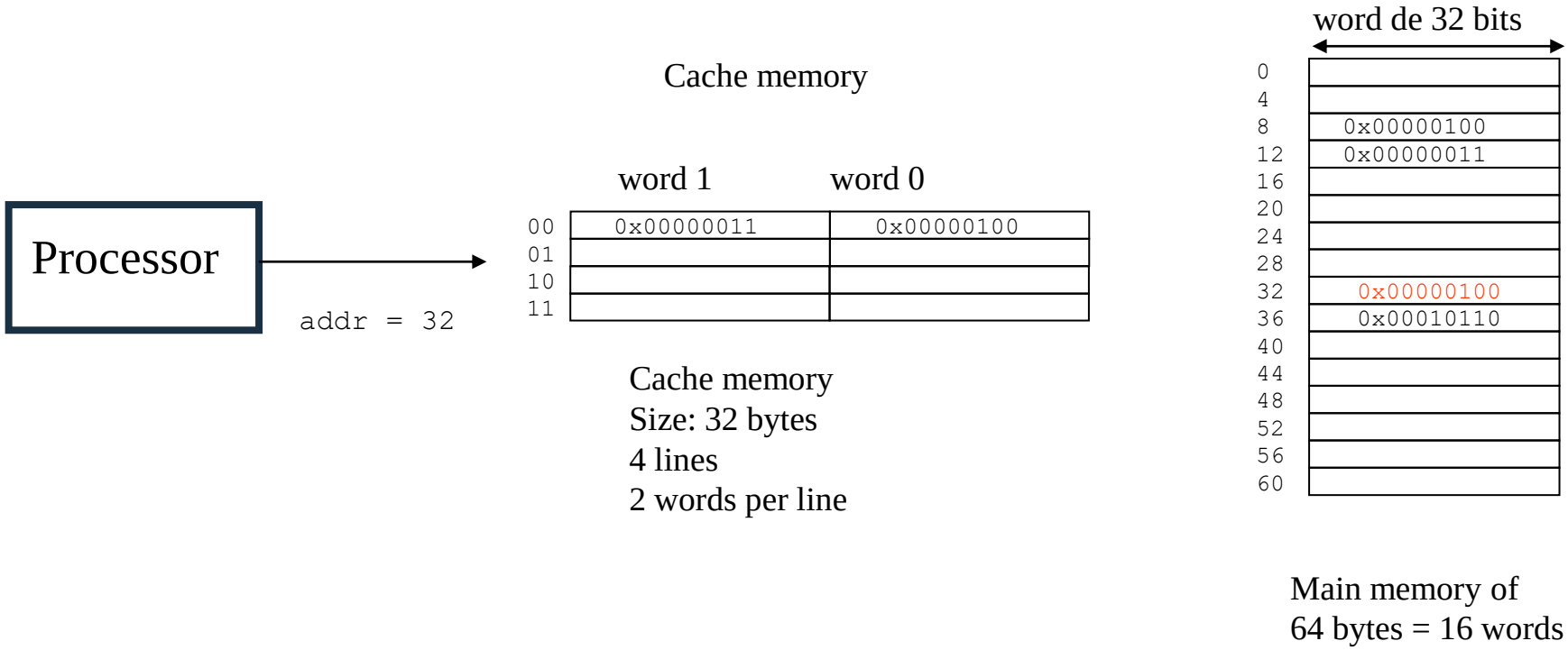
The **block**
is transferred.

Locating a word in the cache



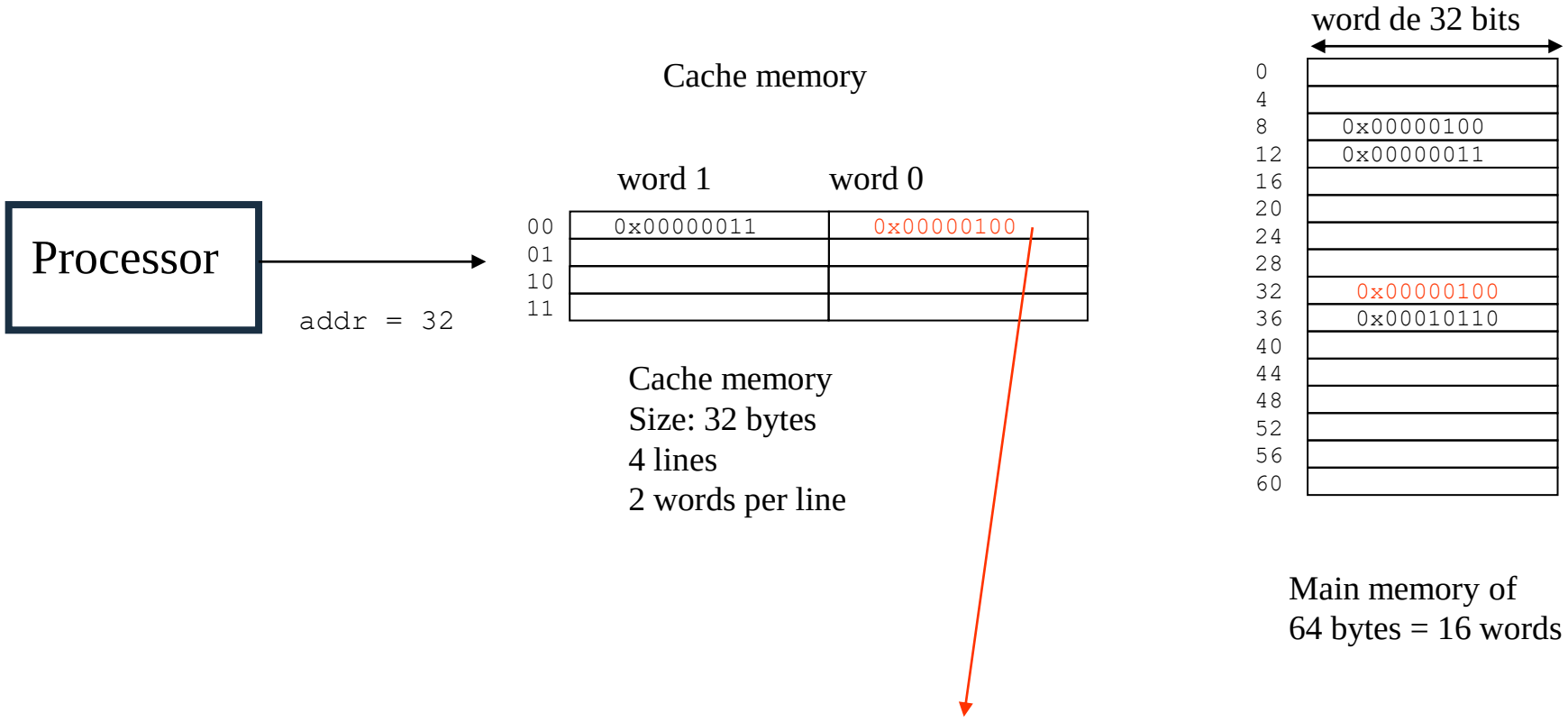
The **word**
is transferred.

Locating a word in the cache



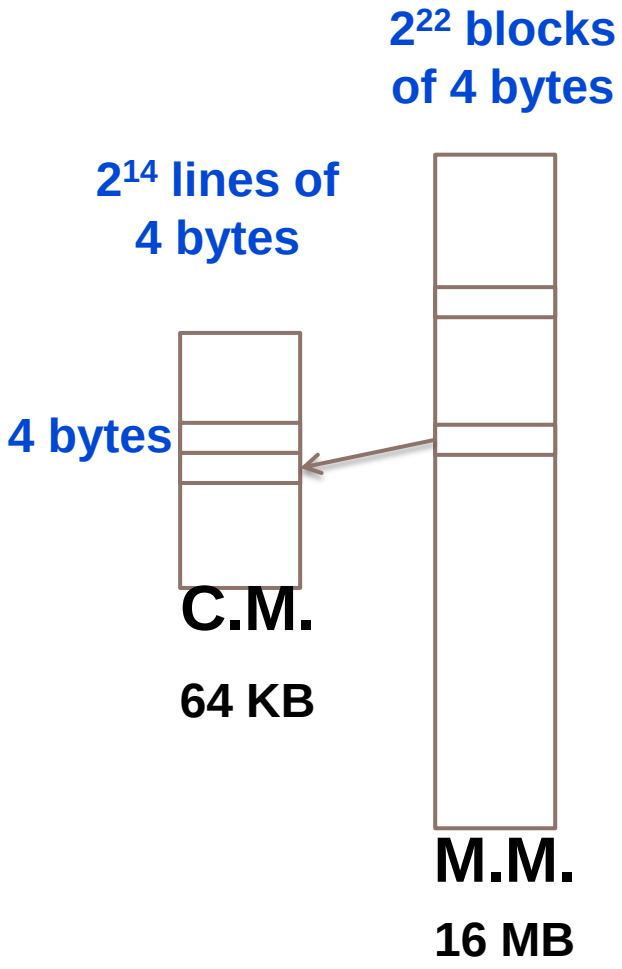
How do I know if it is in the cache?

Locating a word in the cache



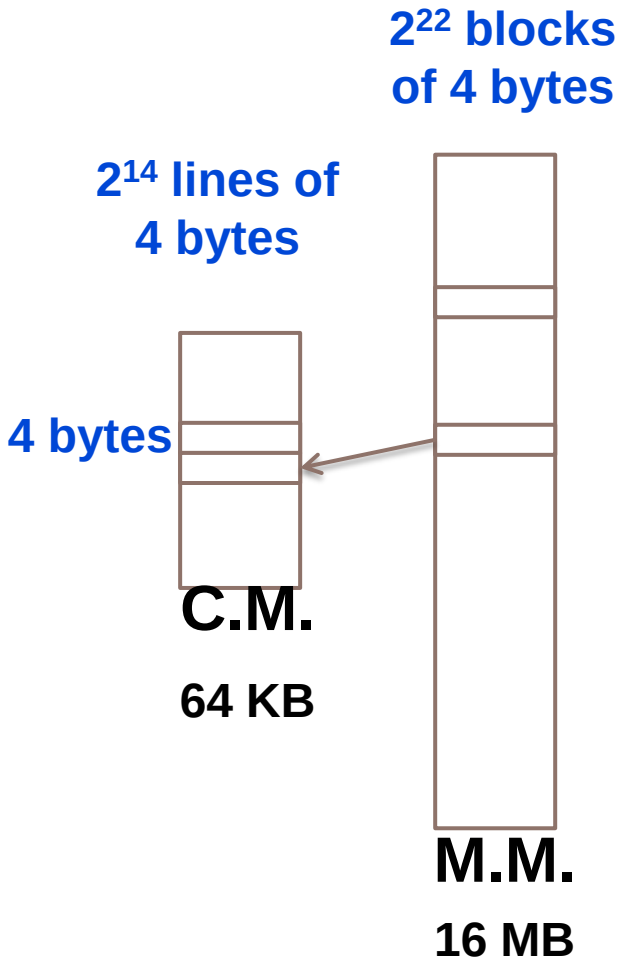
It is the content of address 8 and not 32.

Cache structure and design



- M.M. and C.M. are divided into blocks of equal size.
- Each M.M. block will have a corresponding C.M. line (block in cache)

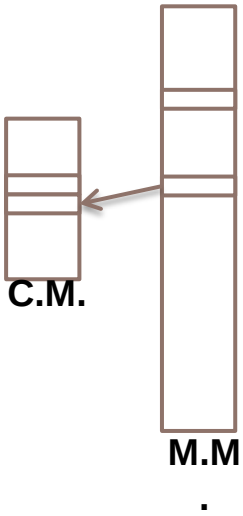
Cache structure and design



- M.M. and C.M. are divided into blocks of equal size.
- Each M.M. block will have a corresponding C.M. line (block in cache)
- The design determines:
  - Size
  - Mapping function
  - Replacement Algorithm
  - Write policy
- Different designs for L1, L2, ... are common.

Cache size

- The total size and the size of the lines into which it is organized
- Determined by studies on widely used codes

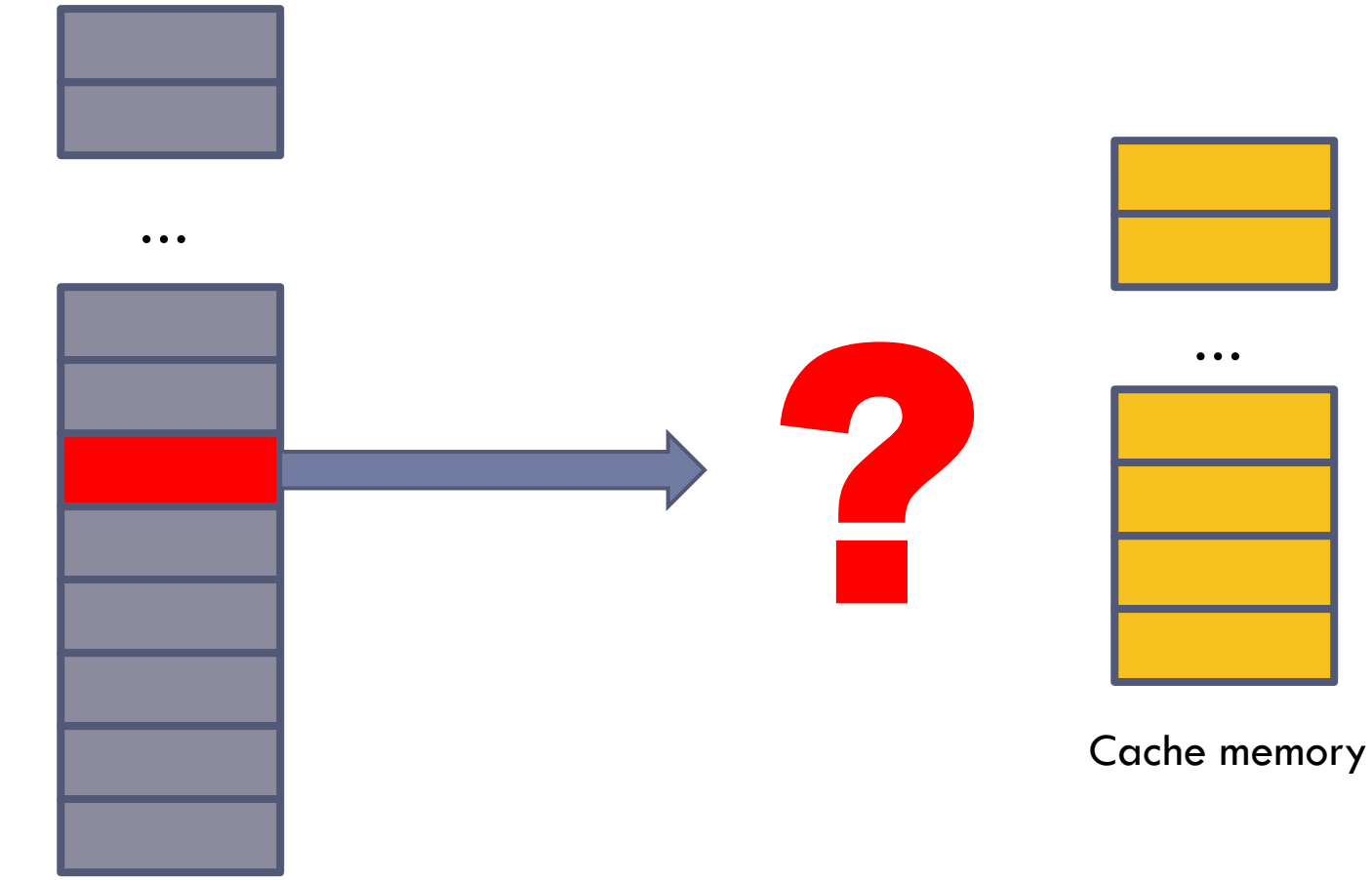


Mapping function

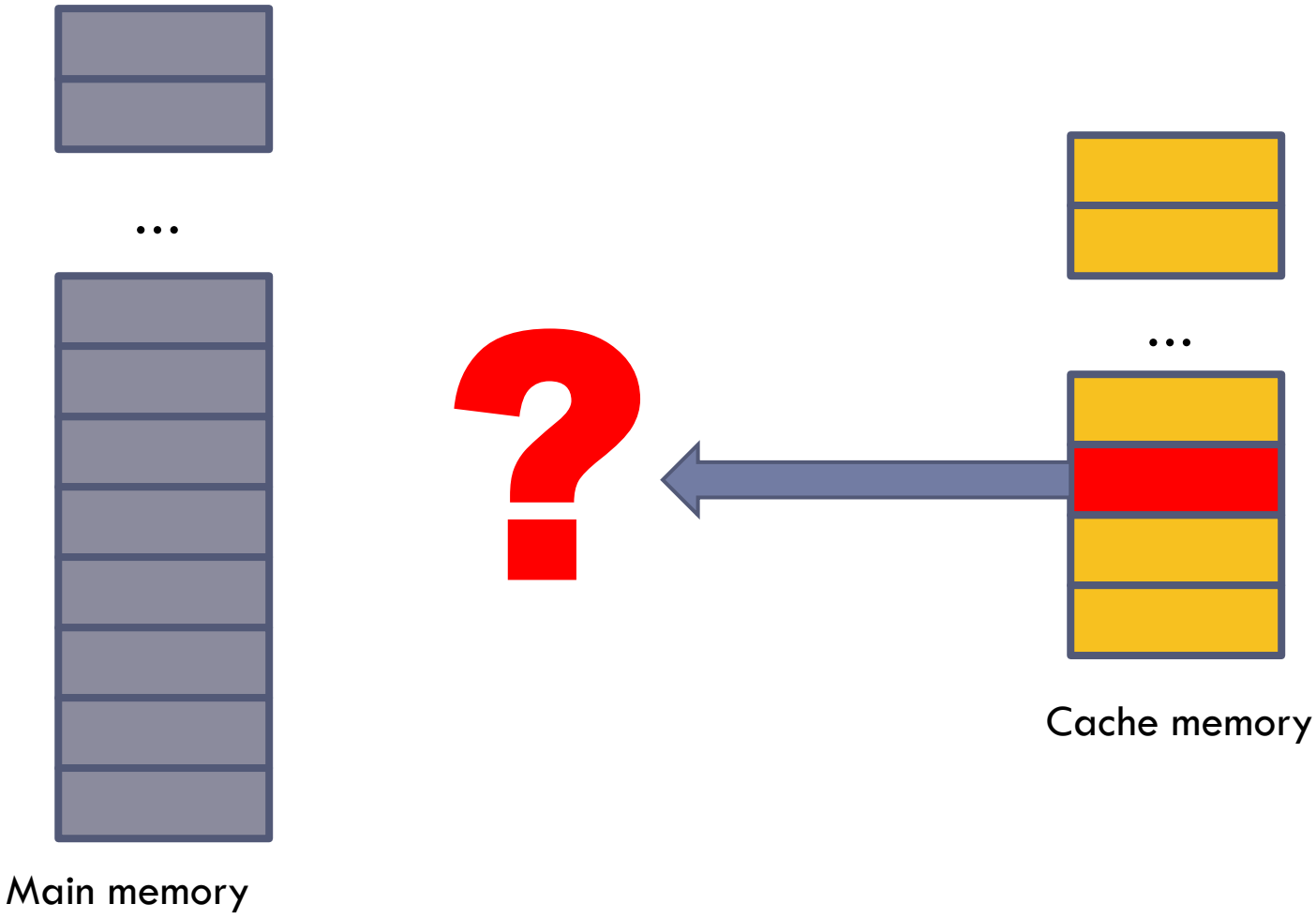
- Algorithm that determines where in the Cache memory a specific block of the Main memory can be stored.

- A mechanism that allows to know which specific block of Main memory is in a line of the Cache memory.
  - Labels are associated with the lines

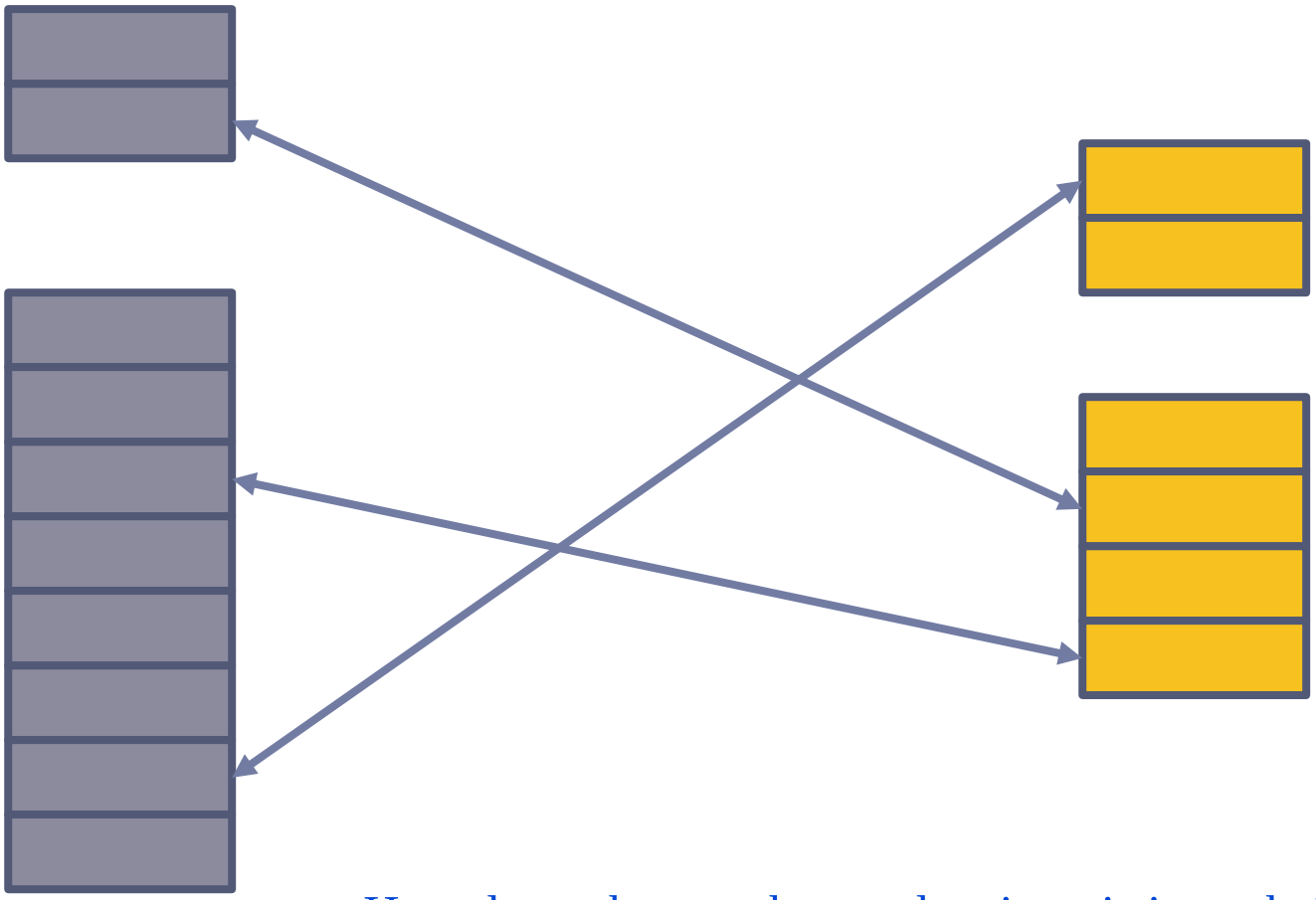
Location in cache memory



Location in main memory



Mapping function

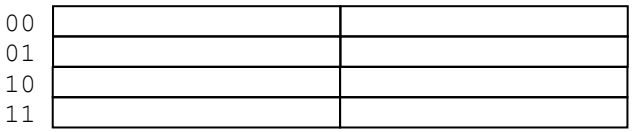


How do we know where a data item is in cache?

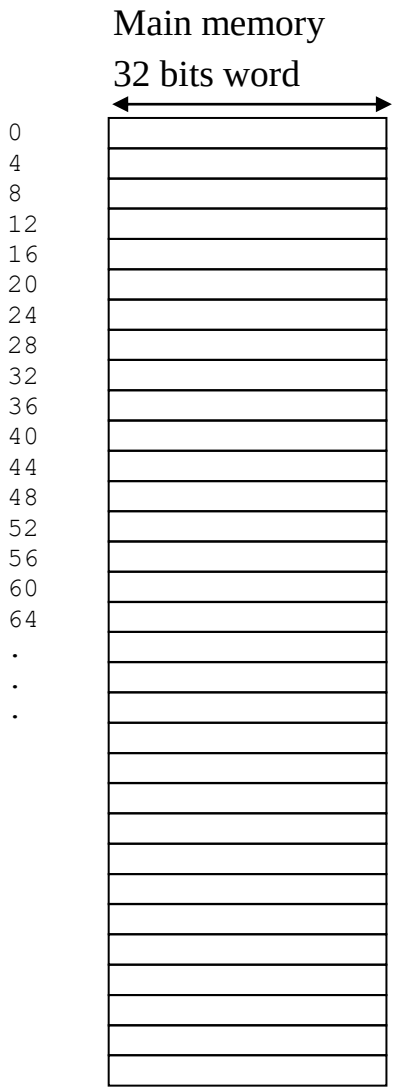
Mapping functions

- Direct mapping function
- Associative mapping function
- Set associative mapping function

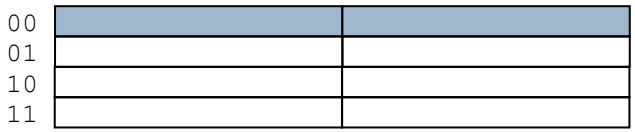
Direct mapped



Cache memory
Size: 32 bytes
4 lines
2 words per line

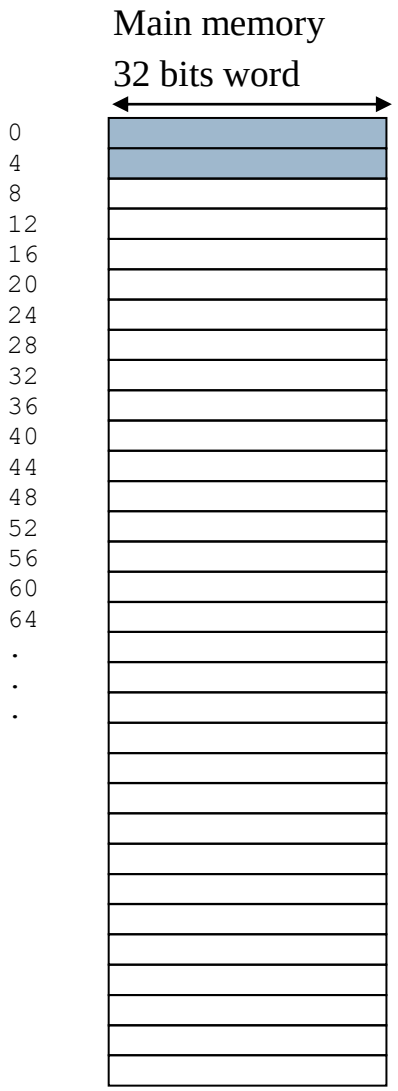


Direct mapped

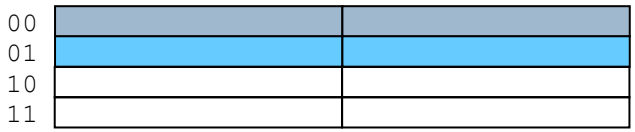


Block 0 - line 0

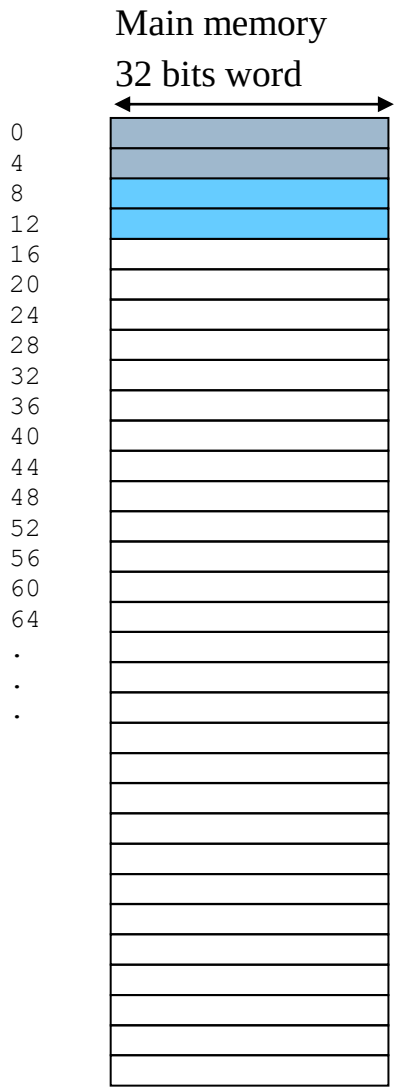
Cache memory
Size: 32 bytes
4 lines
2 words per line



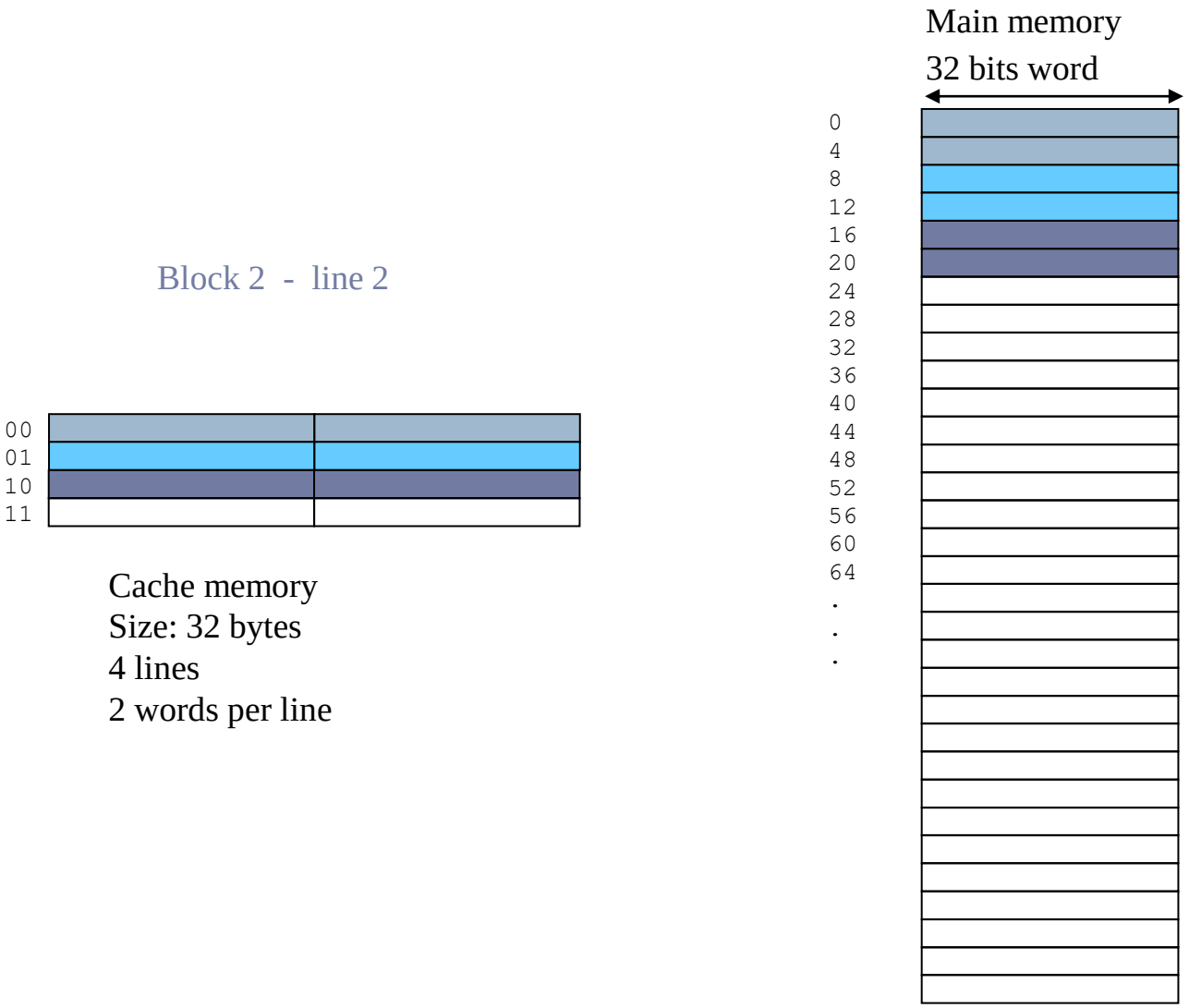
Direct mapped



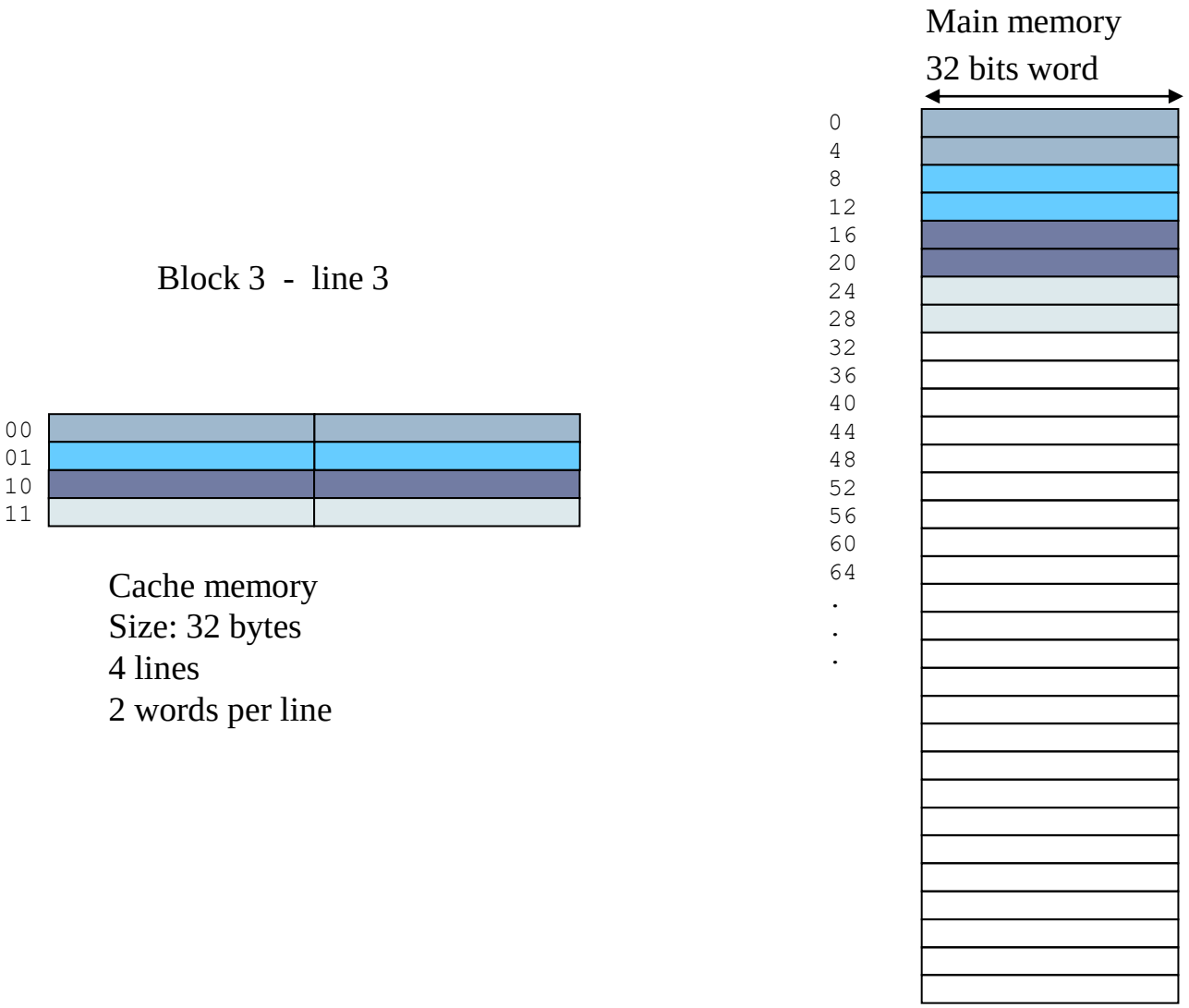
Cache memory
Size: 32 bytes
4 lines
2 words per line



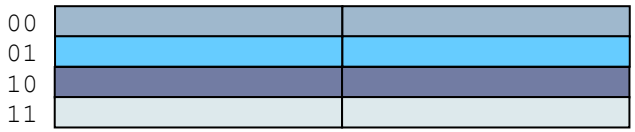
Direct mapped



Direct mapped

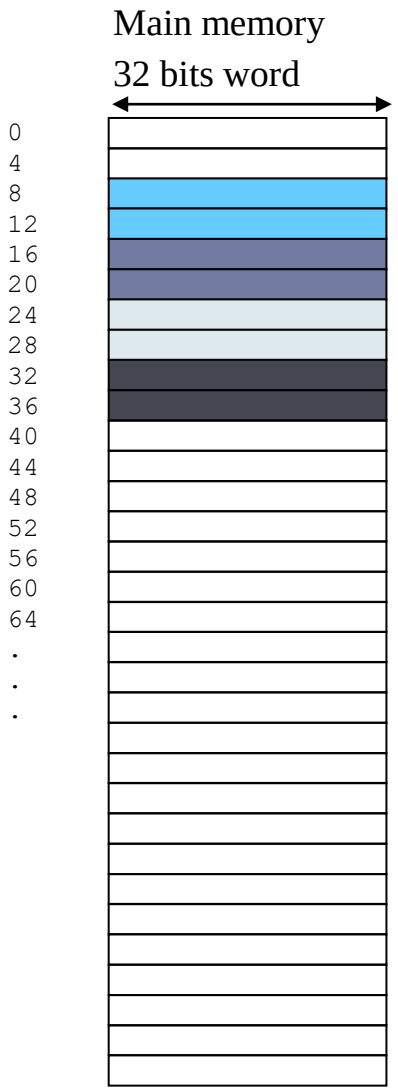


Direct mapped



Block 4 - line 0

Cache memory
Size: 32 bytes
4 lines
2 words per line

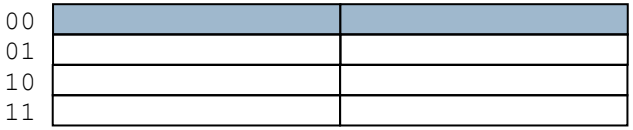


Direct mapped

- In general:
  - The K memory block is stored in the line:

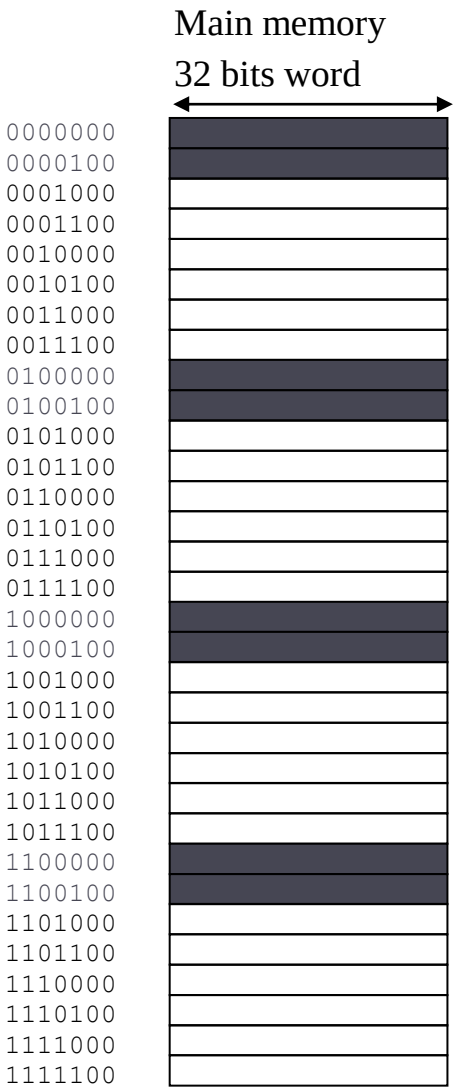
    K mod <number of lines>

Direct mapped

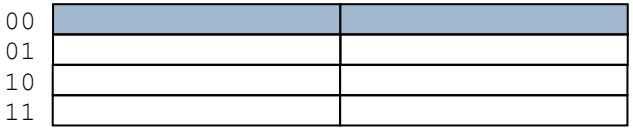


Cache memory
Size: 32 bytes
4 lines
2 words per line

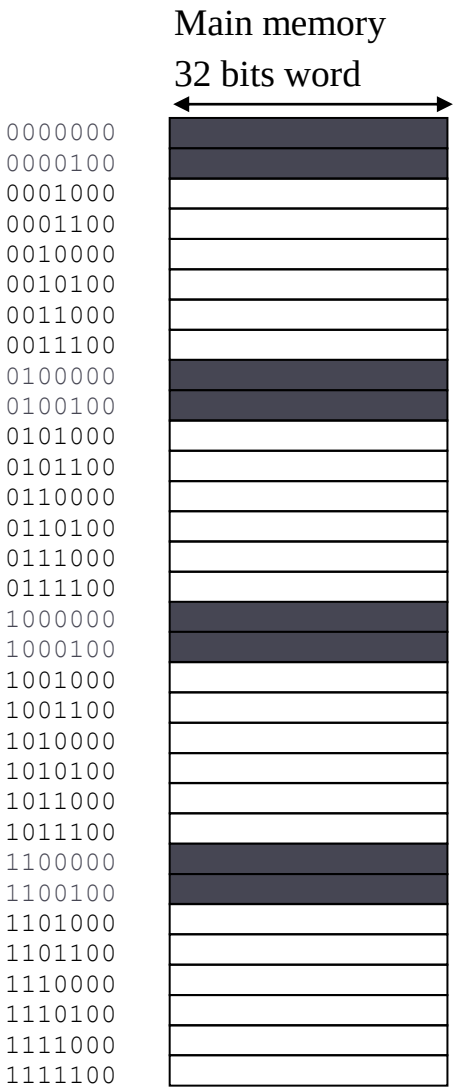
Several blocks in the same line



Direct mapped

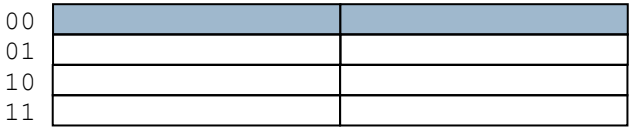


Cache memory
Size: 32 bytes
4 lines
2 words per line

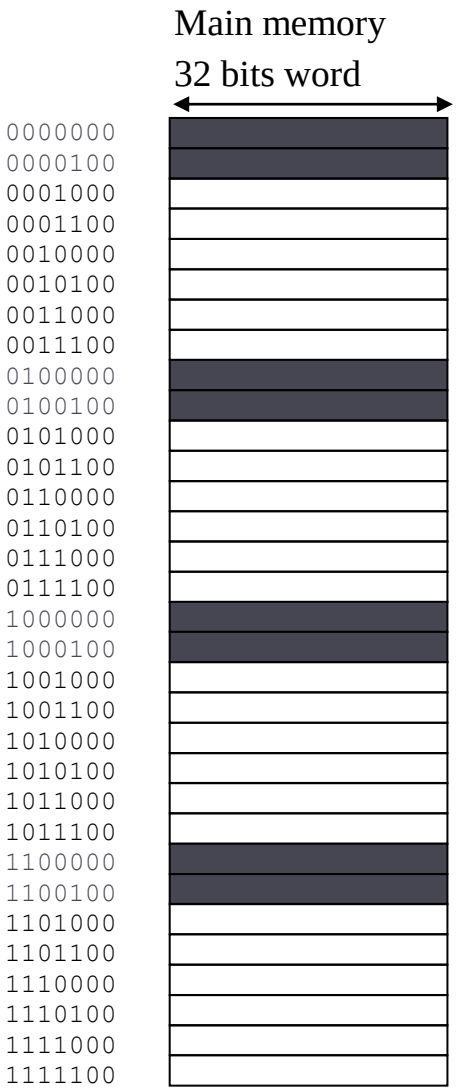


How do we know which memory block is stored in a line?
Example: the address 0100100

Direct mapped



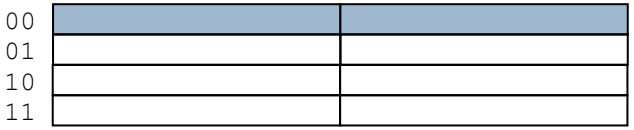
Cache memory
Size: 32 bytes
4 lines
2 words per line



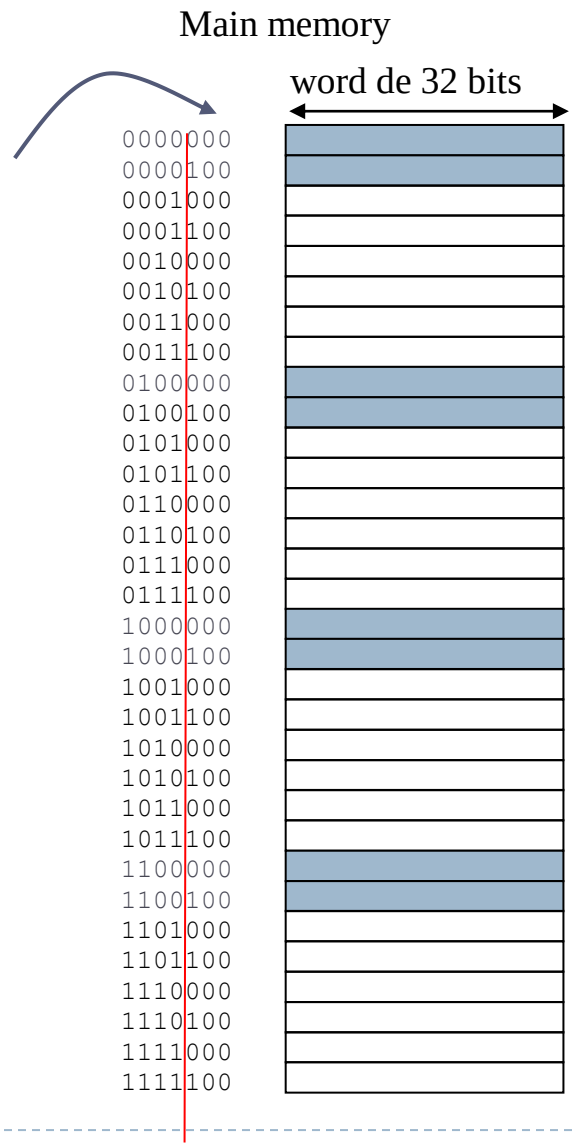
How do we know which memory block is stored in a line?
Example: the address 0100100
A label is added to each line

Direct mapped
Idea

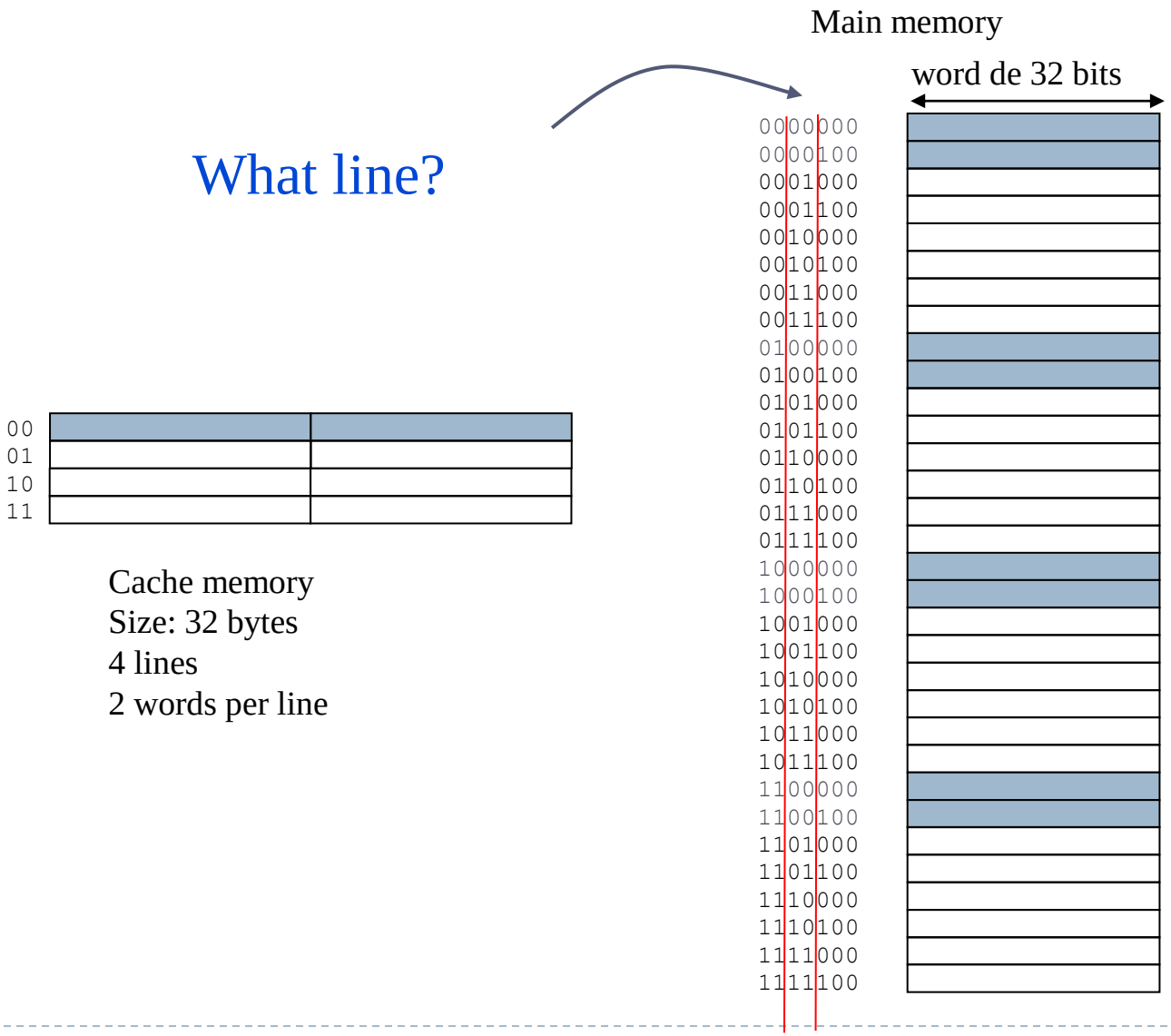
which byte inside the line?
8-byte lines



Cache memory
Size: 32 bytes
4 lines
2 words per line

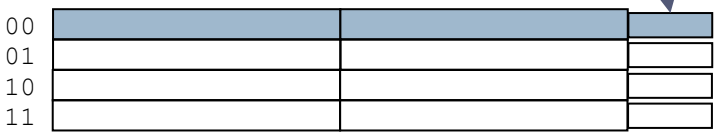


Direct mapped Idea

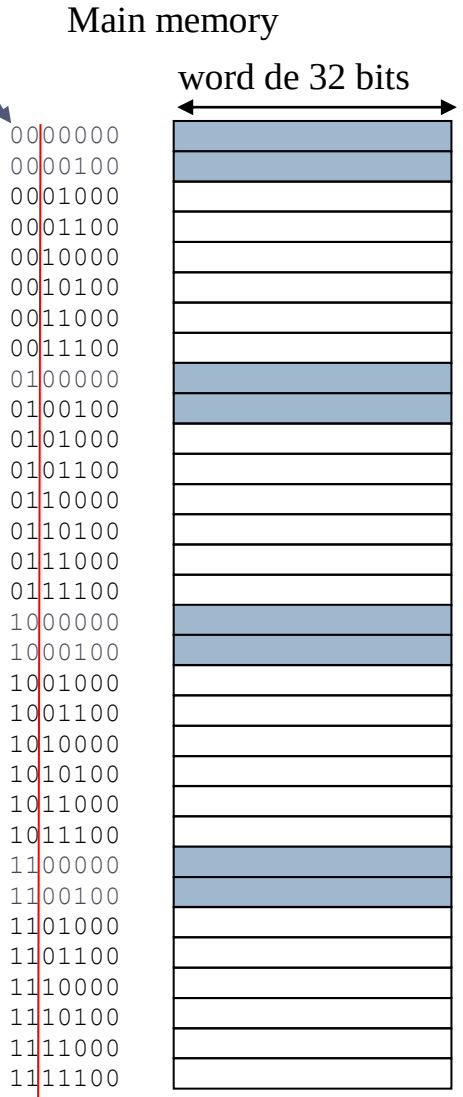


Direct mapped
Idea

label associated to the line that differentiates the blocks that go to the same line

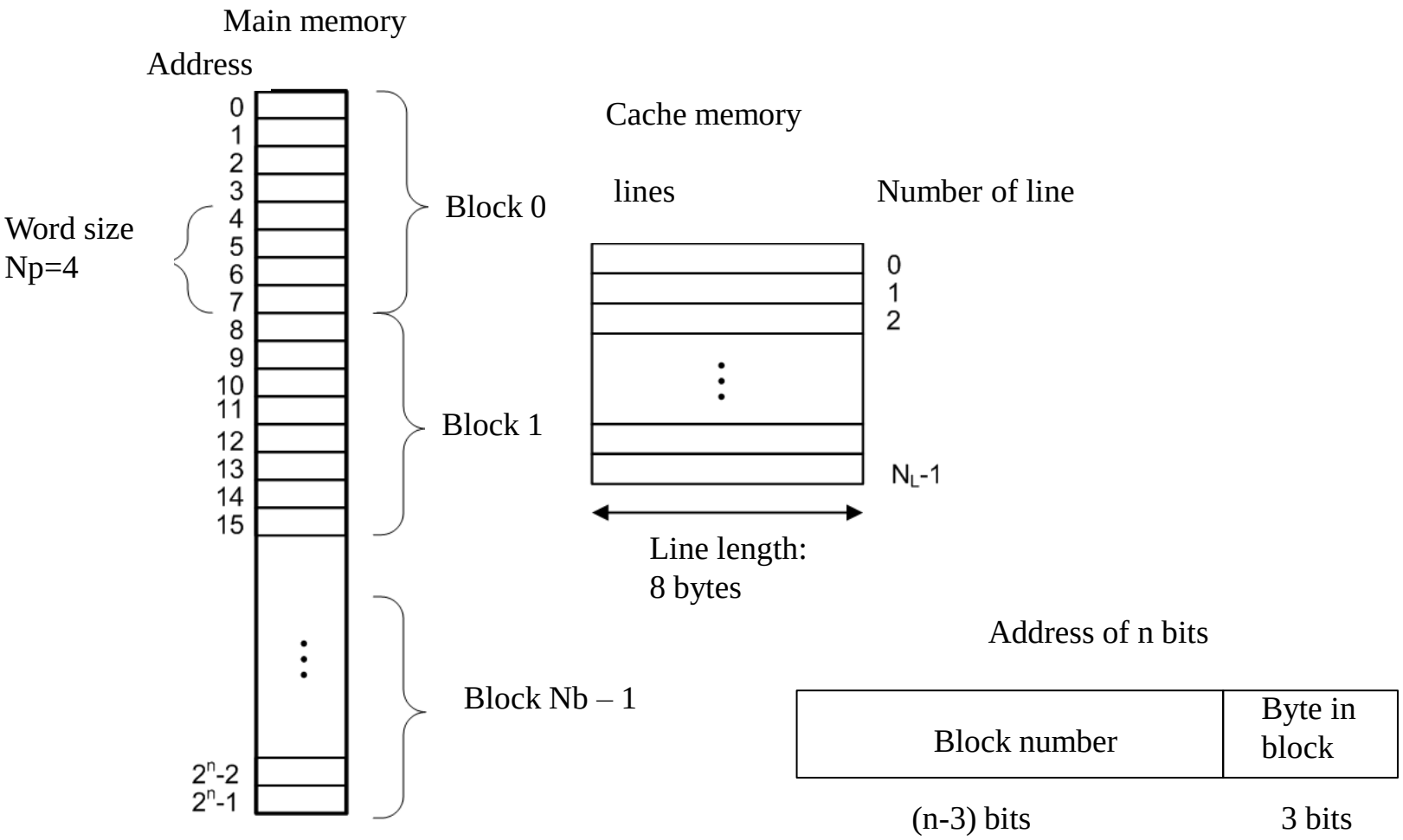


Cache memory
Size: 32 bytes
4 lines
2 words per line

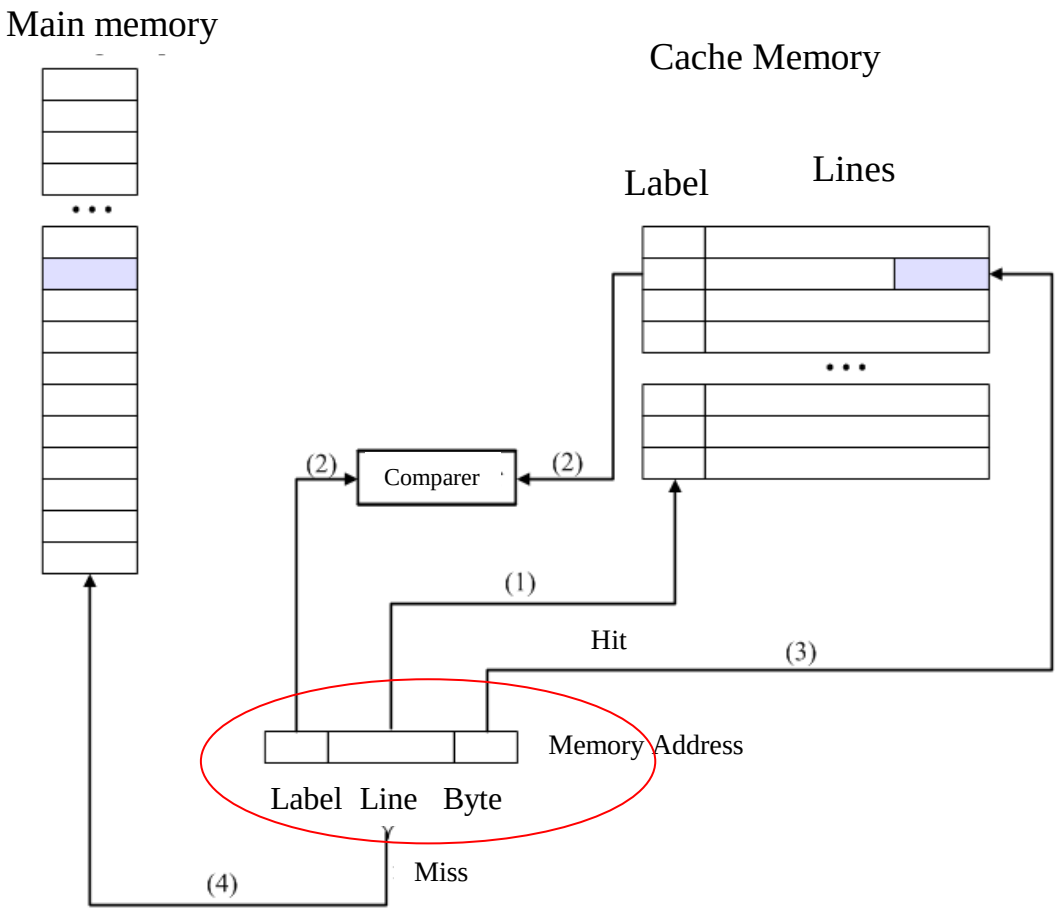


The tag (upper bits of the address) is also stored in the cache memory.

Example of organization



Scheme for a direct mapped cache memory

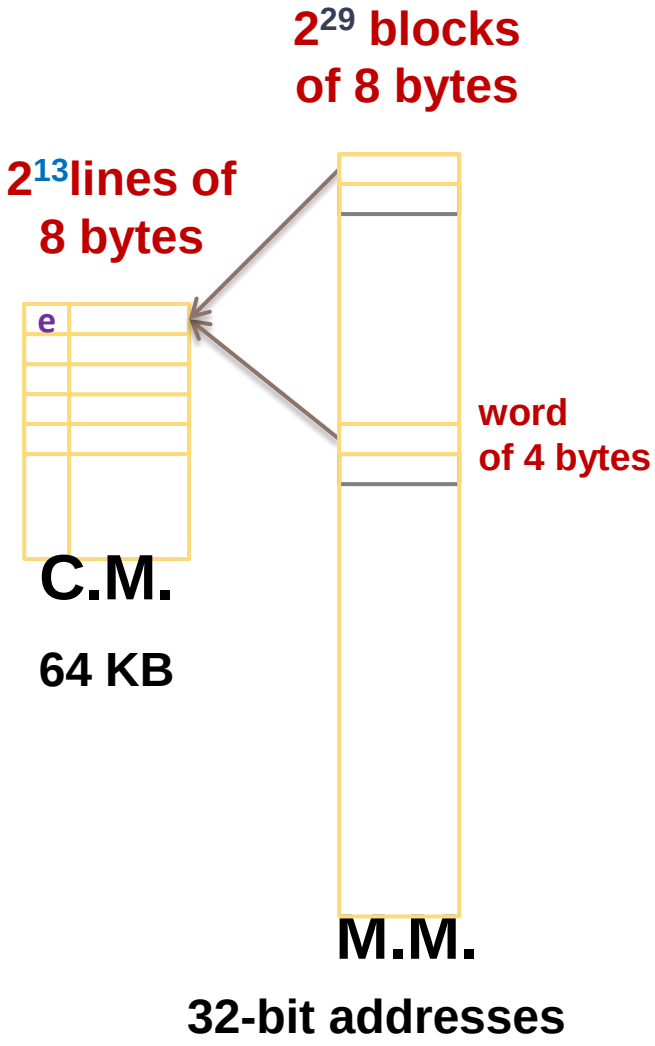


Direct mapping function (example)

- Each block of main memory is associated with a single cache line (always the same)
- Main memory address is interpreted as :

32-16	13	3
label	line	byte

- If in '*line*' there is '*label*', then block is cache (HIT)
- Simple, inexpensive, but can cause many misses depending on access pattern

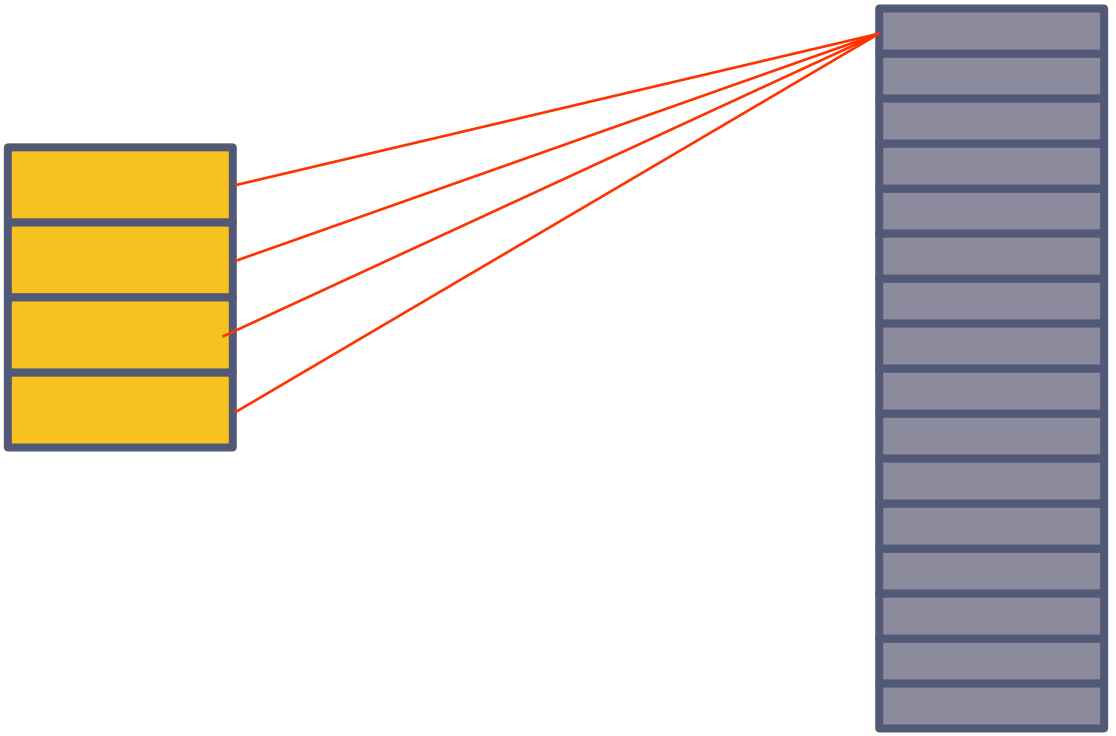


Exercise

- Given a 32-bit computer with a 64 KB cache memory and 32-byte blocks. If direct matching is used:
  - On which line of the cache memory is the word for address 0x0000408A stored?
  - How can it be fetched quickly?
  - On which line of the cache memory is the word for address 0x1000408A stored?
  - How does the cache know if the word stored on that line corresponds to the word at address 0x0000408A or to the word at address 0x1000408A?

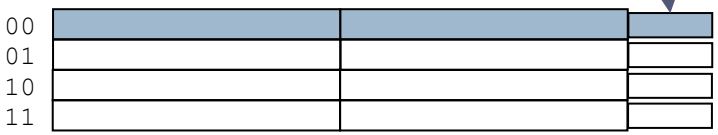
Associative mapping

- Each block of main memory can be stored in any cache line.

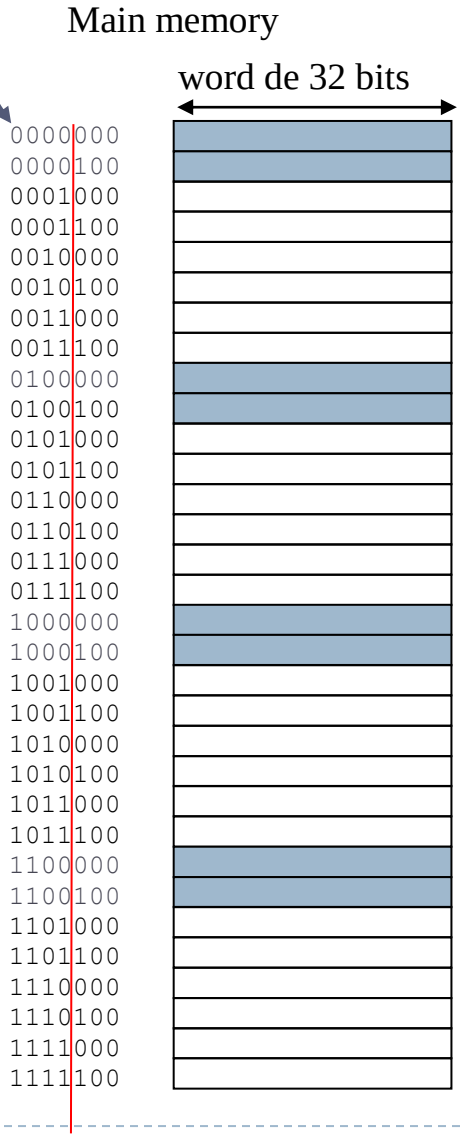


Associative mapping Idea

label associated with the line that differentiates the blocks that go to the same line

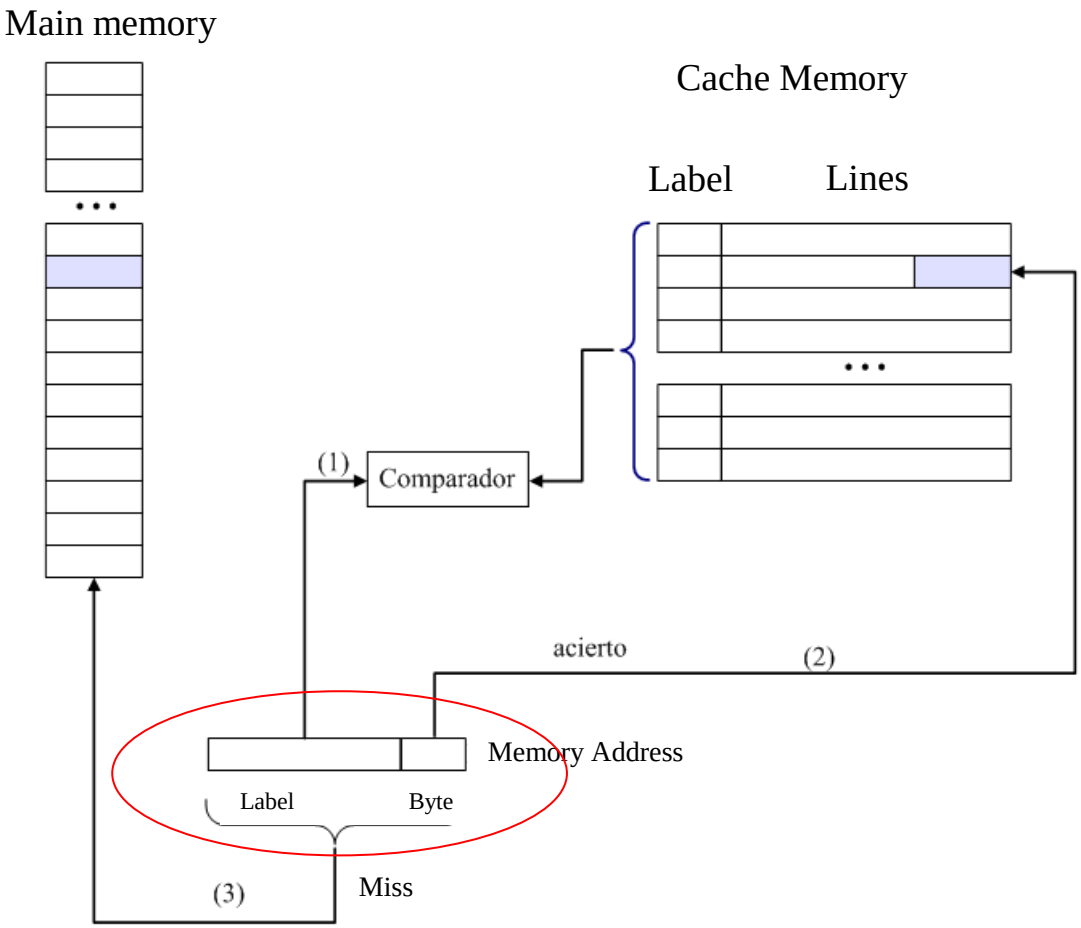


Cache memory
Size: 32 bytes
4 lines
2 words per line



The label (upper bits of the address) is also stored in cache memory

Organization of a Cache memory with associative mapping

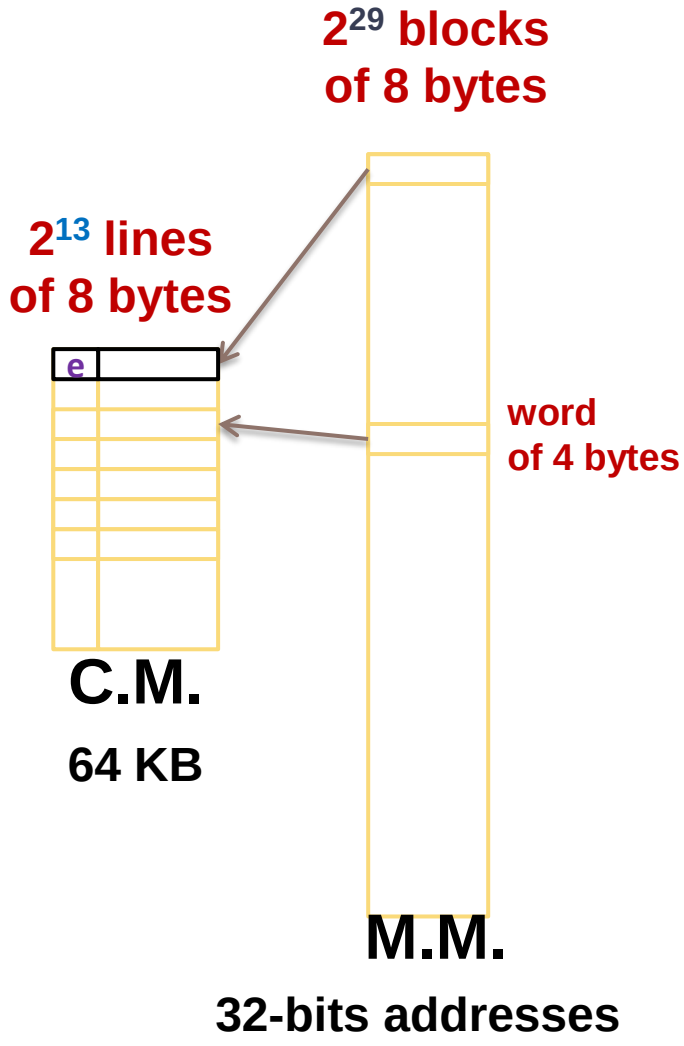


Associative mapping function (example)

- A main memory block can be placed in any cache line
- Main memory address is interpreted as:

32-3	3
label	byte

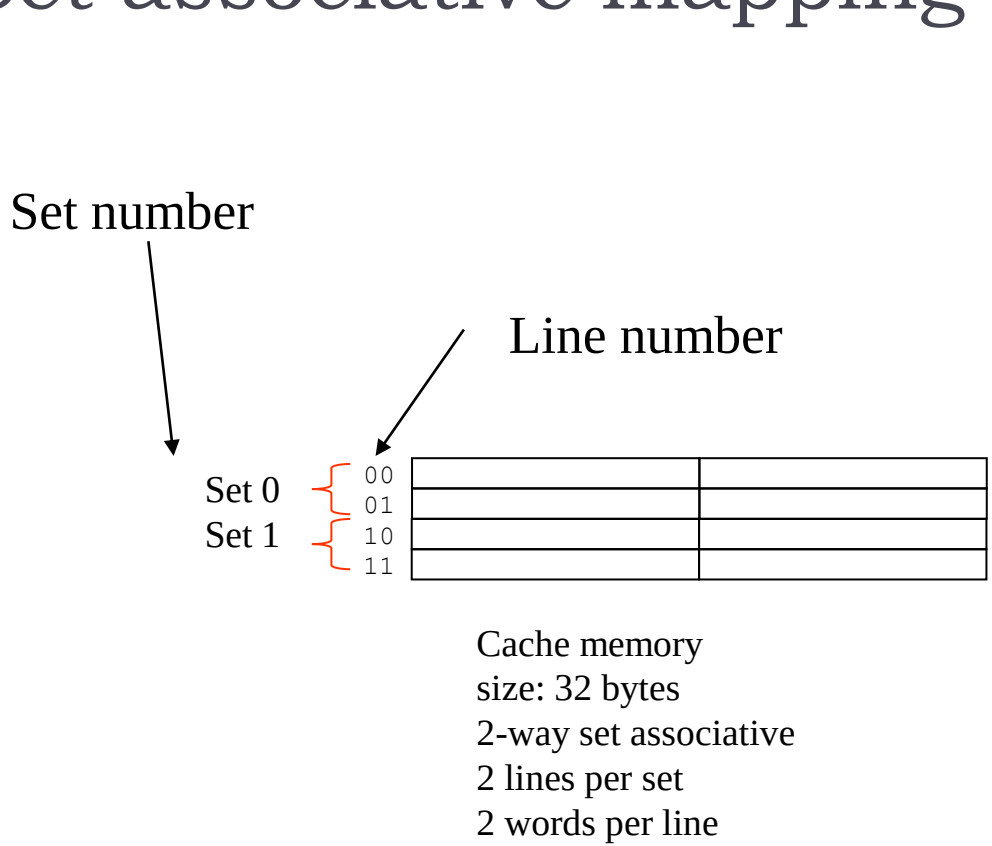
- If there is a line with '*label*' within cache, then block is there
- Access pattern independent, expensive search
- Larger tags: larger caches



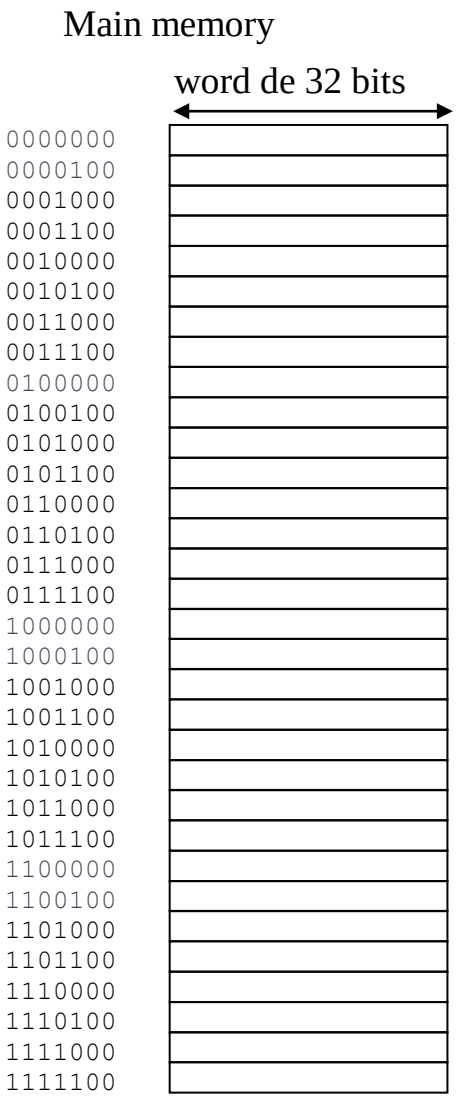
Set associative mapping

- Memory is organized into sets of lines
- One set-associative memory cache of k-ways:
  - Each set stores K lines
- Each block is always stored in the same set
  - Block B is stored in set:
    - B **mod** <number of sets>
- Within a set the block can be stored in any of the lines of that set.

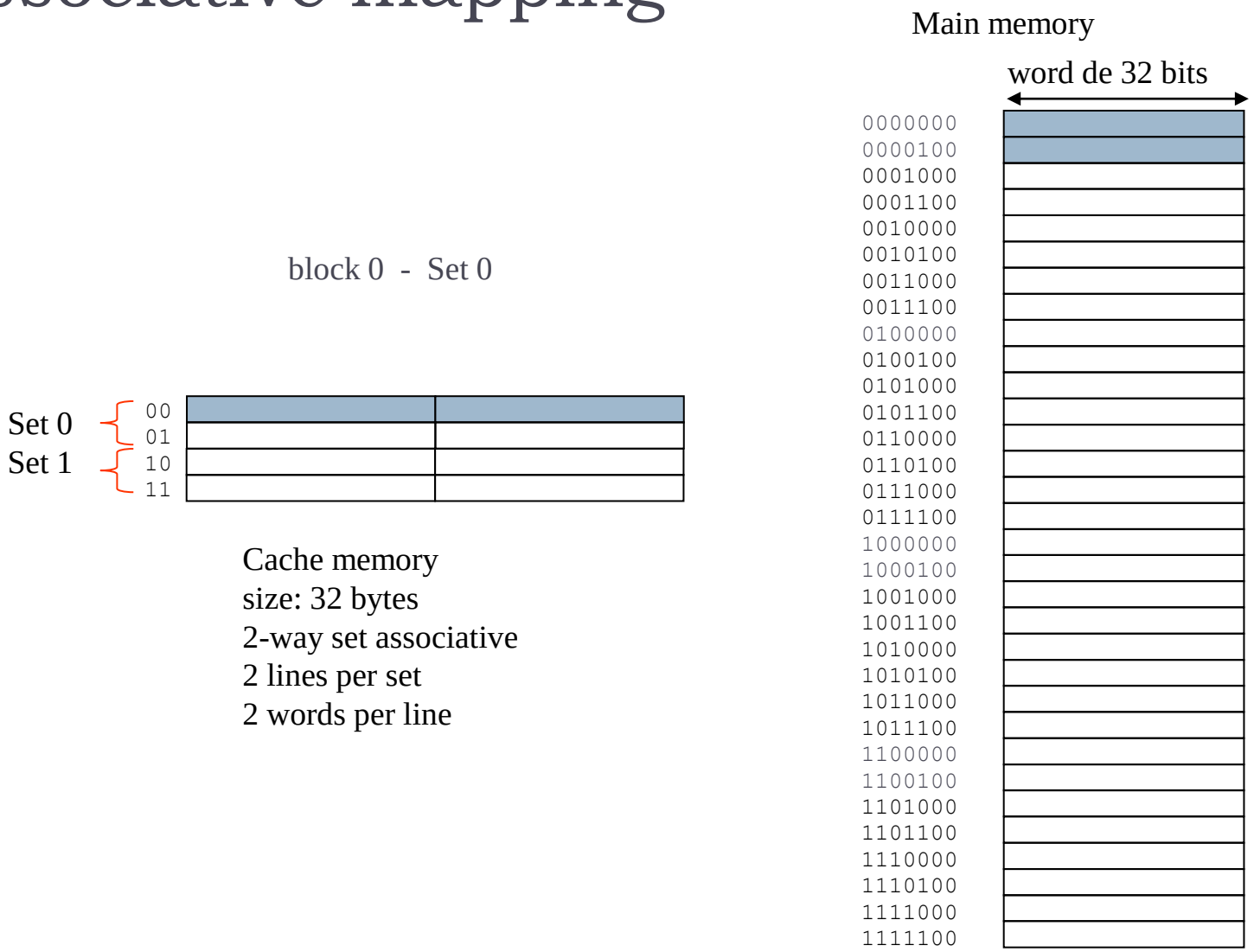
Set associative mapping



Cache memory
size: 32 bytes
2-way set associative
2 lines per set
2 words per line

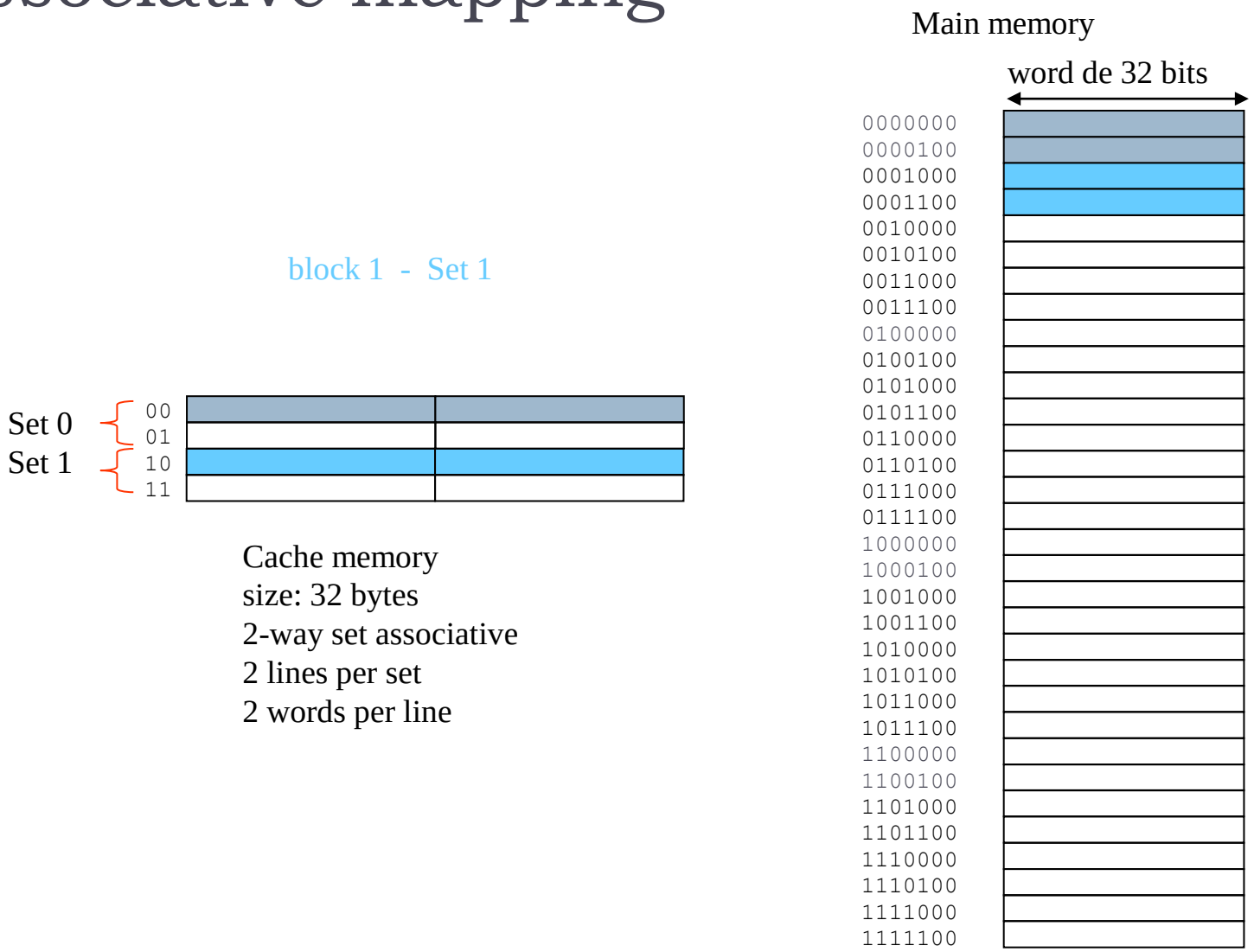


Set associative mapping

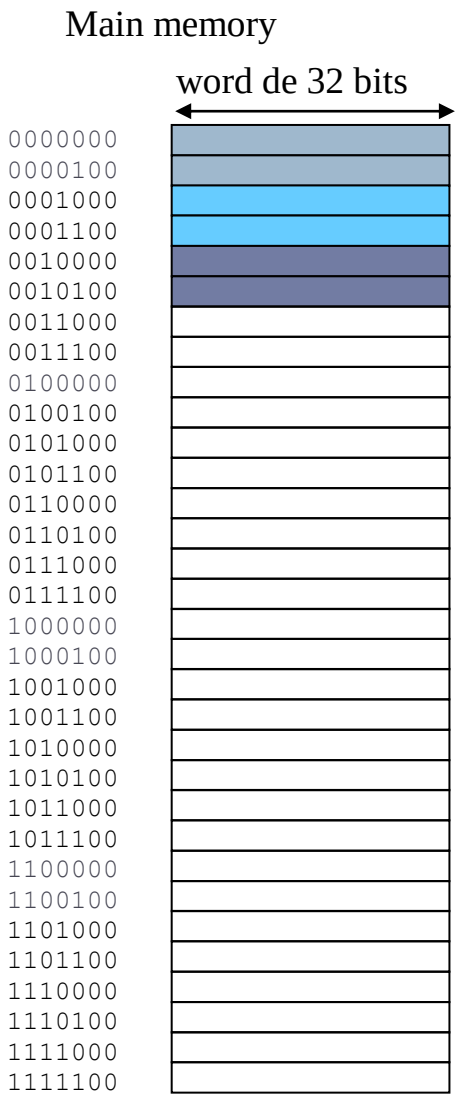


Cache memory
size: 32 bytes
2-way set associative
2 lines per set
2 words per line

Set associative mapping



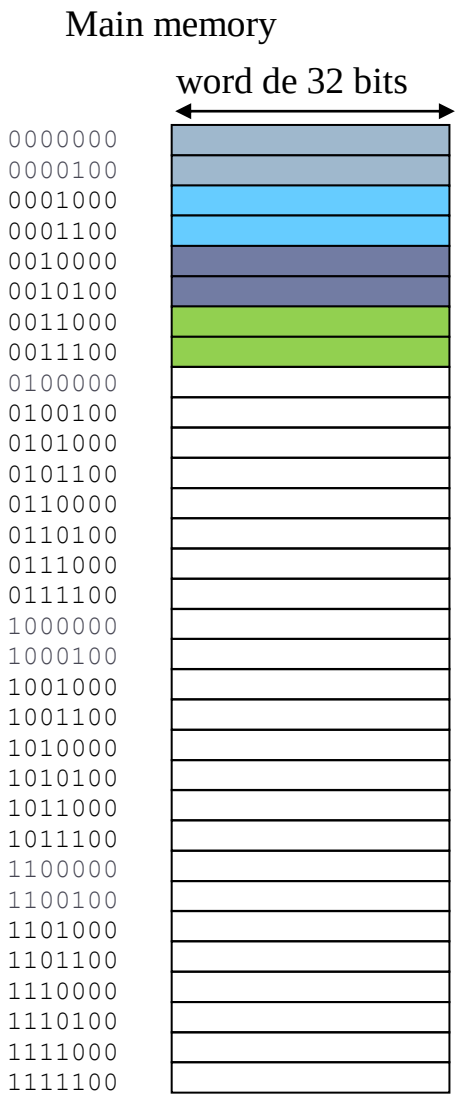
Set associative mapping



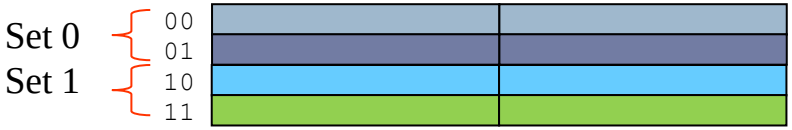
block 2 - Set 0

Cache memory
size: 32 bytes
2-way set associative
2 lines per set
2 words per line

Set associative mapping

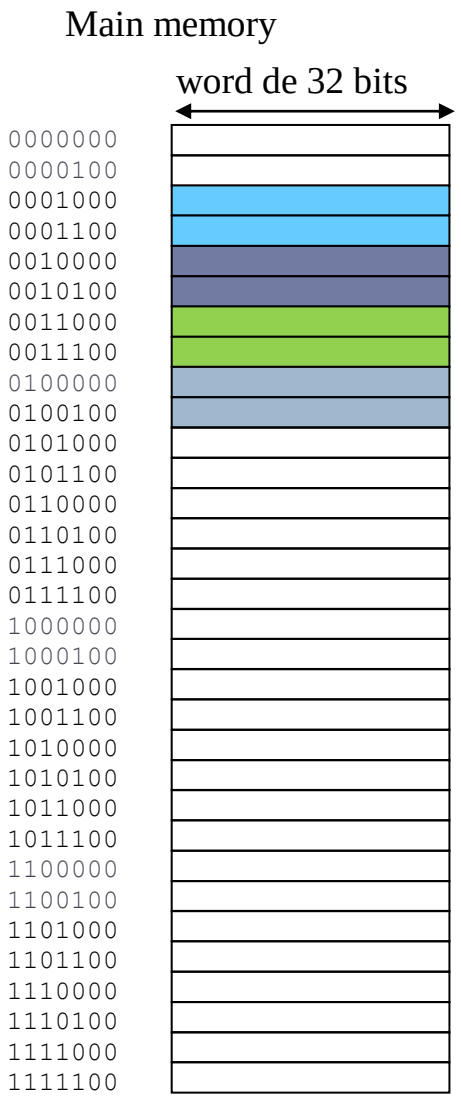


block 3 - Set 1

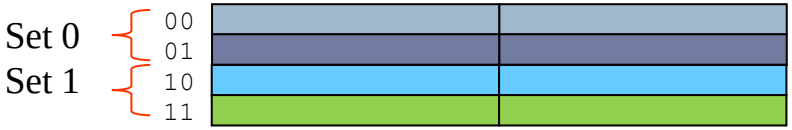


Cache memory
size: 32 bytes
2-way set associative
2 lines per set
2 words per line

Set associative mapping



block 4 - Set 0



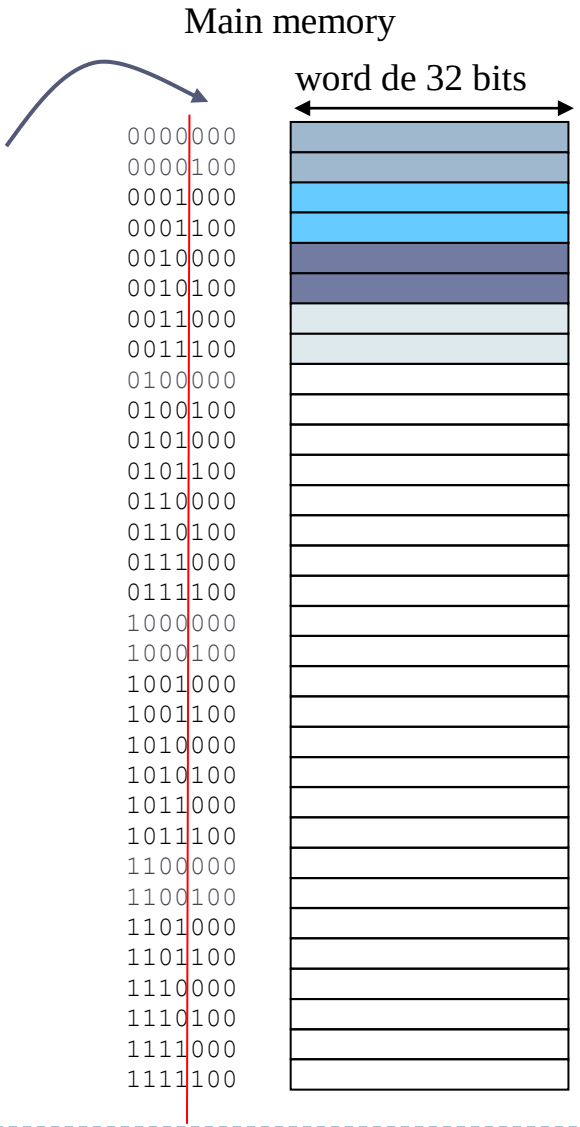
Cache memory
size: 32 bytes
2-way set associative
2 lines per set
2 words per line

We have to discard the line stored before

Set associative mapping

which byte inside the line?
8-byte lines

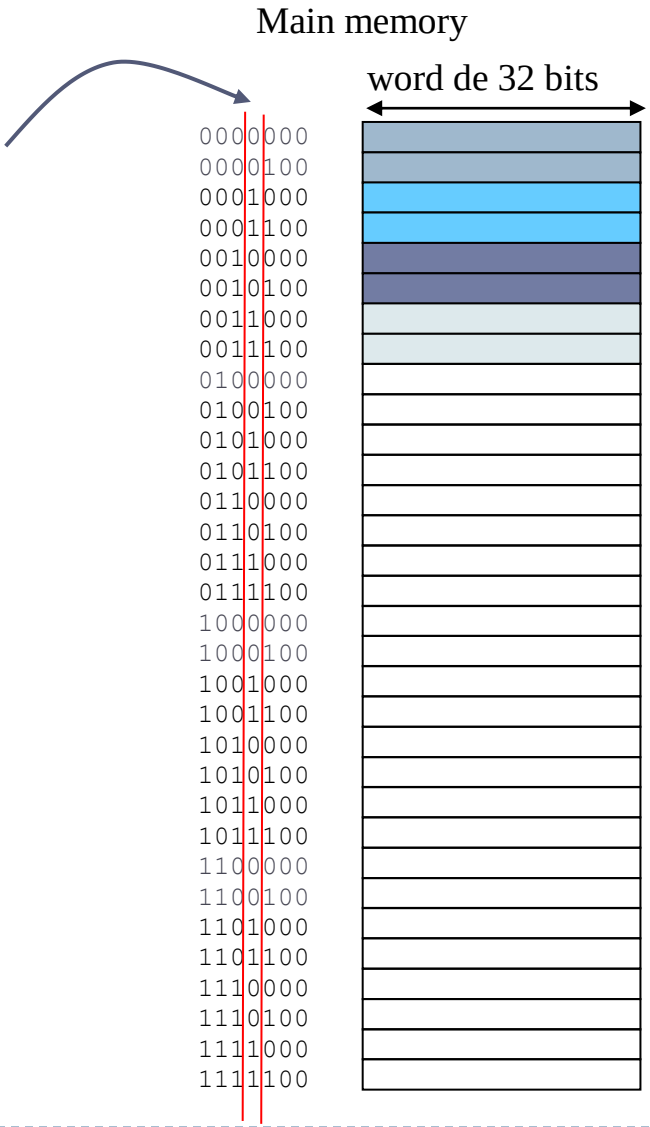
Cache memory
Size: 32 bytes
Set-associative of 2 ways
2 lines per set
2 words per line



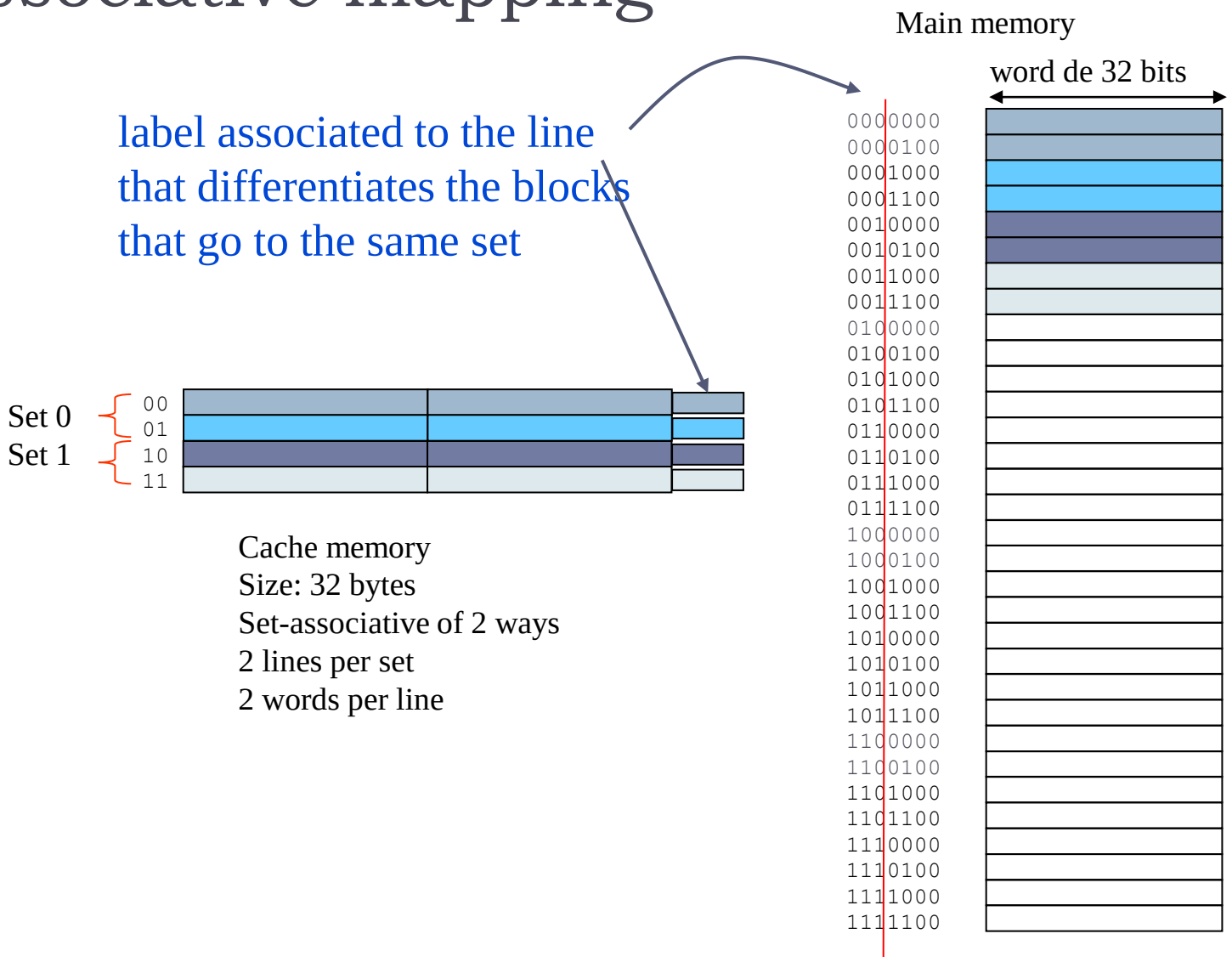
Set associative mapping

which set?
within the set to any line

Cache memory
Size: 32 bytes
Set-associative of 2 ways
2 lines per set
2 words per line



Set associative mapping

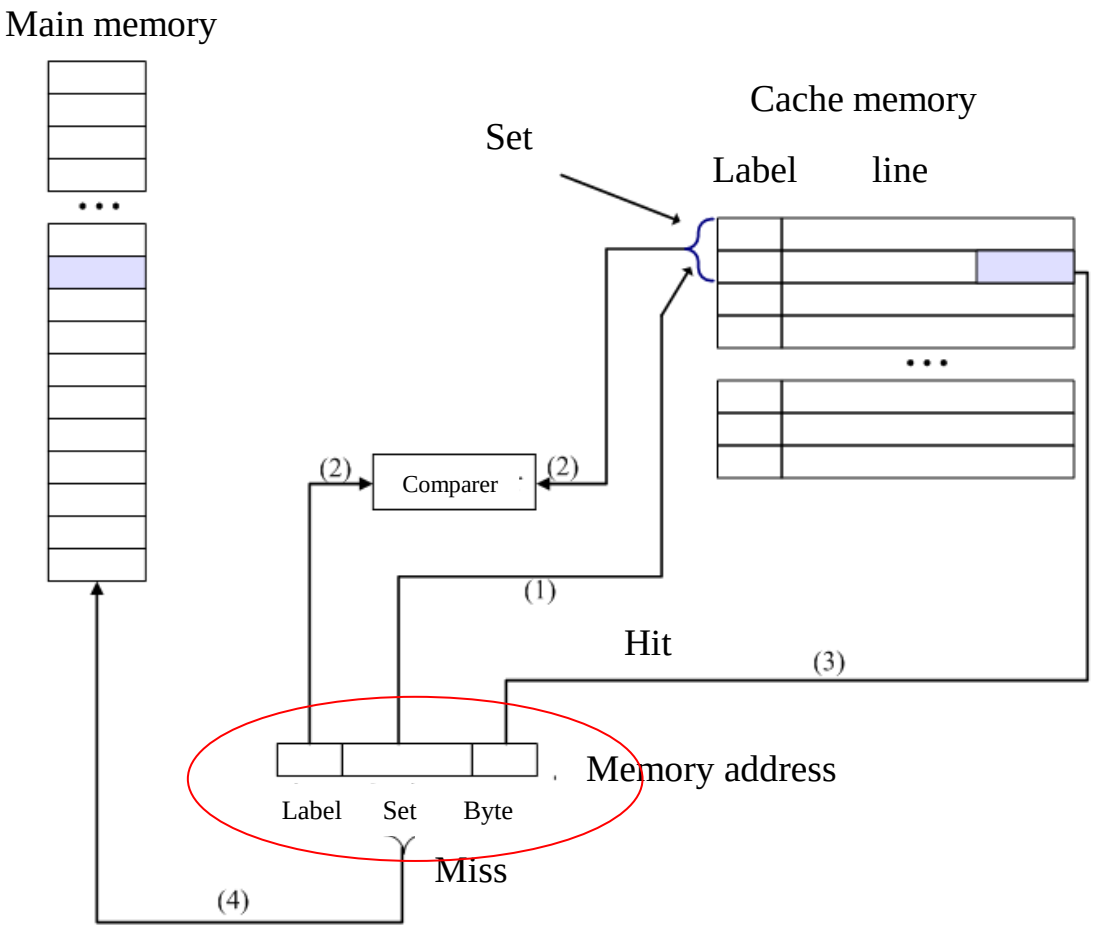


Cache memory
Size: 32 bytes
Set-associative of 2 ways
2 lines per set
2 words per line

Set associative mapping

- Establishes a compromise between flexibility and cost:
  - It is more flexible than direct correspondence.
  - It is less expensive than associative matching.

Scheme for a set associative cache memory

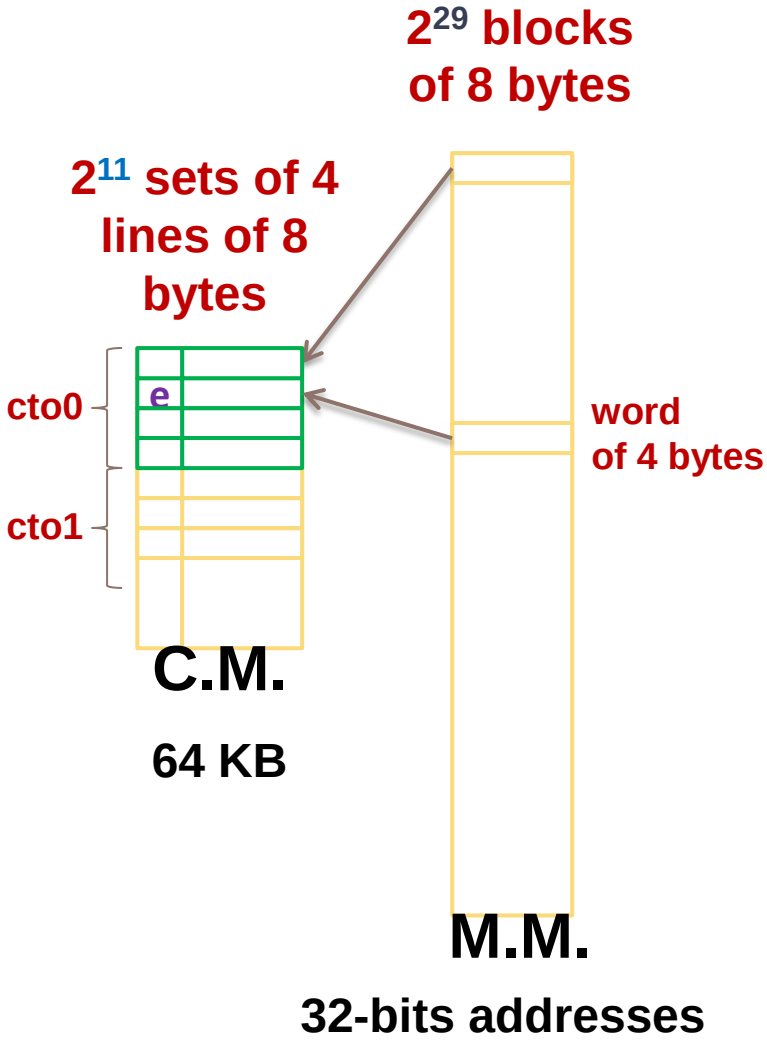


Set-associative mapping (example)

- **Set-associative**:
  - A block of main memory can be placed in any cache line (way) of a given set.
  - Main memory address is interpreted as:

18	11	3
label	set	word

  - If there is a line with '*label*' in the set '*set*', then there is the block in cache
  - [A] the best of direct and associative
    [D] (less) expensive search



Block replacement

- When all cache entries contain blocks from main memory (MM):
  - It is necessary to select a line to be left free in order to bring a block from the MM.
    - Direct: no possible choice
    - Associative: select a line from the cache.
    - Set-associative: select a line from the selected set.
  - There are several algorithms for selecting the cache line to be released.

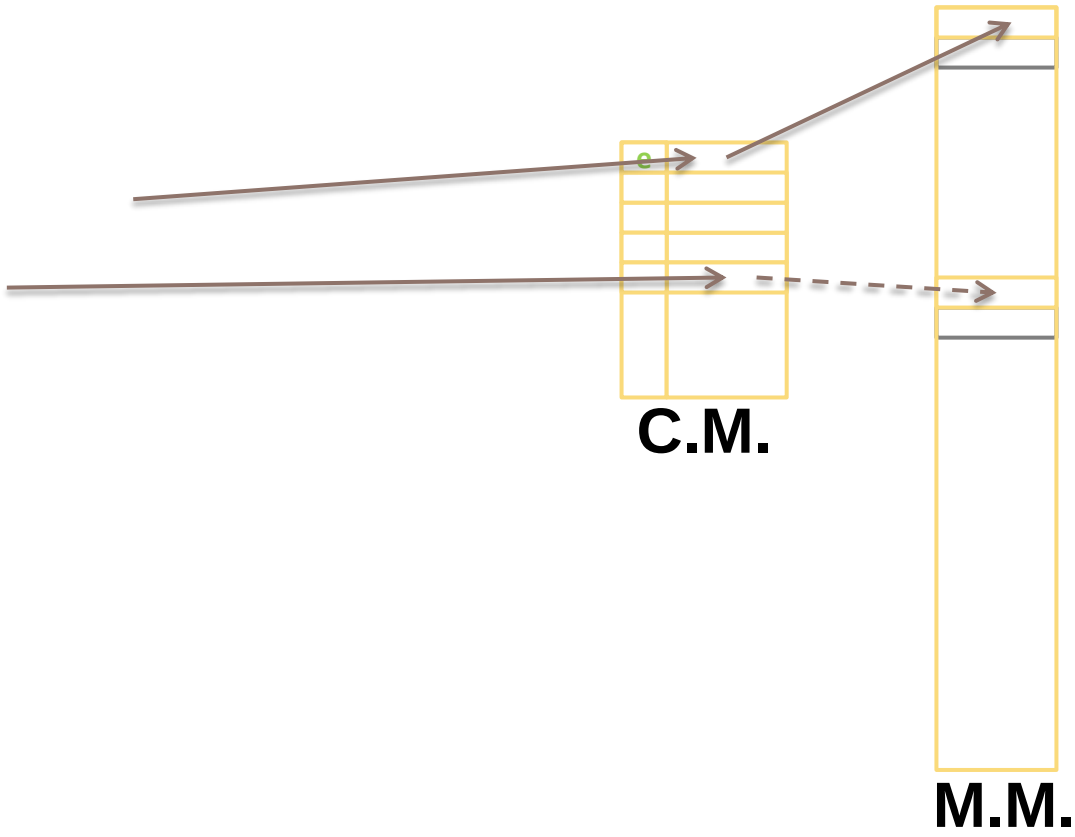
Replacement algorithms

- FIFO
  - *First-in-first-out*
  - Replaces the line that has been in the cache the longest.
- LRU:
  - *Least Recently Used*
  - Replaces the line that has not been used for the longest time with no reference to it.
- LFU:
  - *Least Frequently Used*
  - This replaces the candidate line in the cache that has had the fewest references.

Write policy

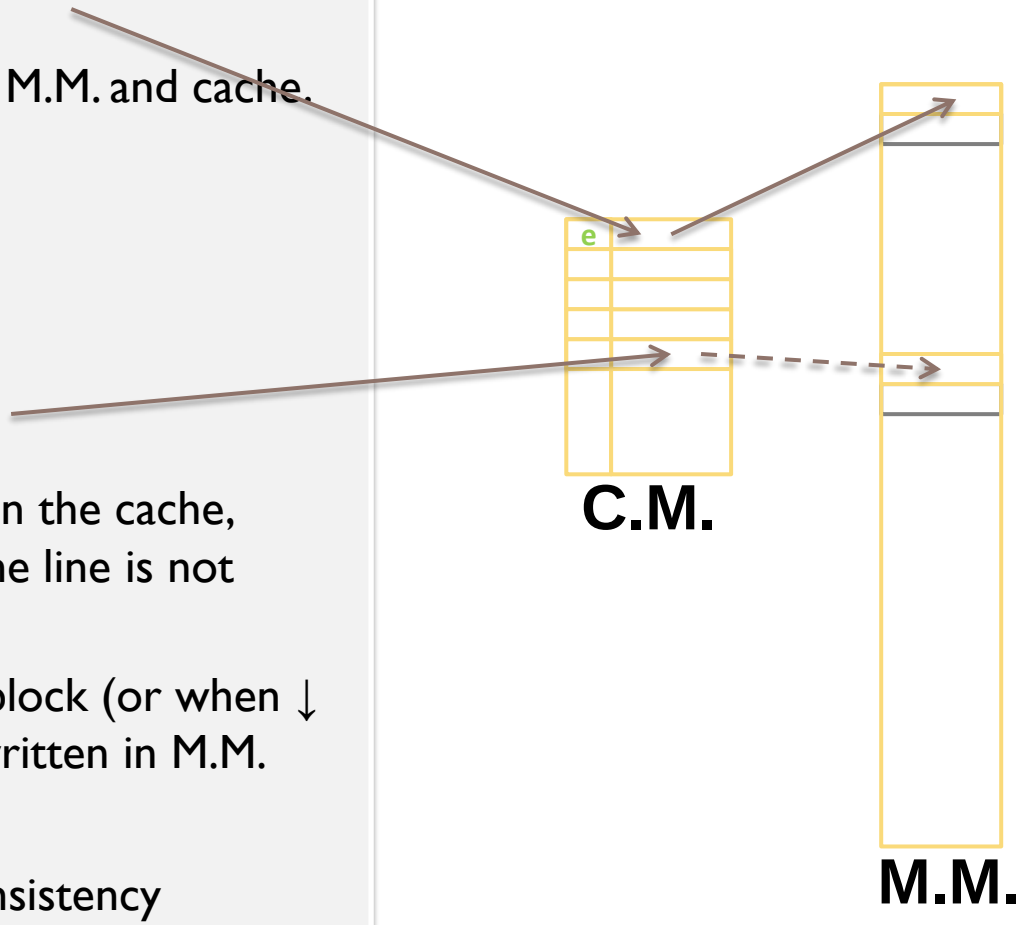
- When a data is modified in Cache memory, the Main memory must be updated at some point.

- Techniques:
  - Write through.
  - Write back.



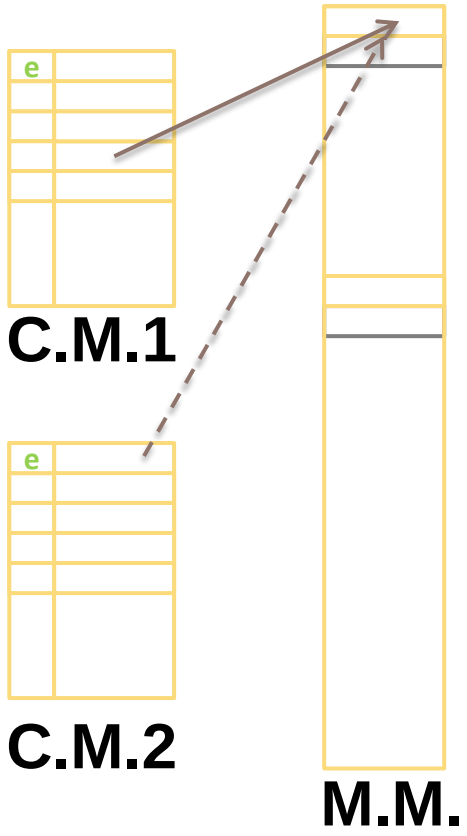
Write policy

- Write through:
  - Writing is done in both M.M. and cache.
  - [A] Coherence
  - [D] Heavy traffic
  - [D] Slow writing
- Write back:
  - The write is only done in the cache, indicating in a bit that the line is not flushed in M.M.
  - When substituting the block (or when ↓ traffic with M.M.) it is written in M.M.
  - [A] Speed
  - [D] Coherence + inconsistency

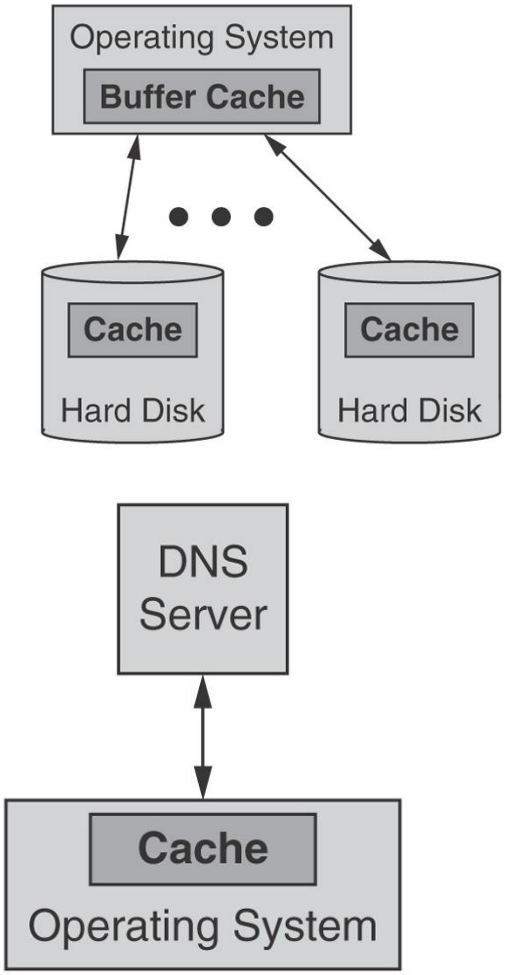


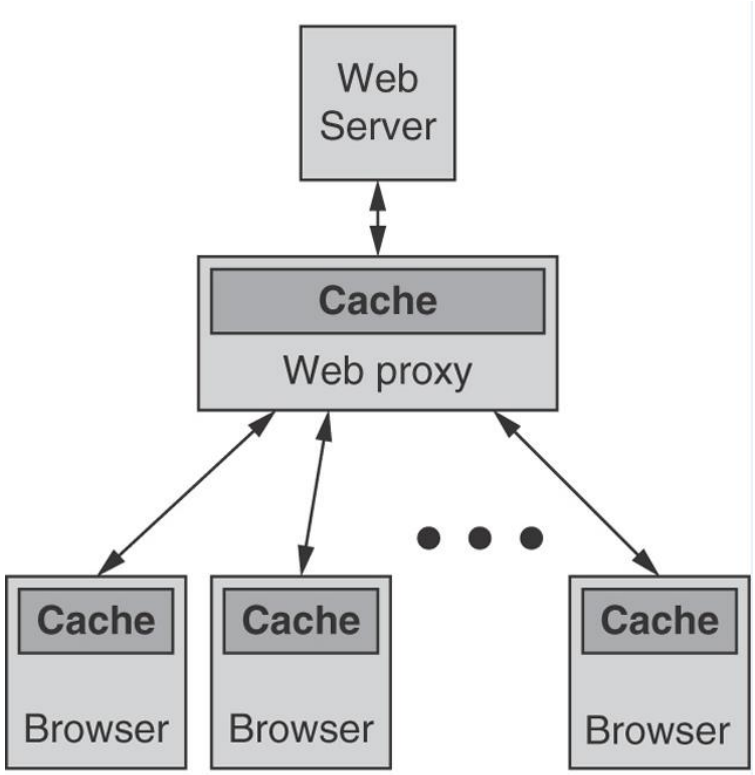
Write policy

- E.g.: Multicore CPU with per-core cache
  - Cached writes are only seen by one core
  - If each core writes to the same word, what is the final result?
- E.g.: I/O module with DM.
  - Updates by DMA (direct memory access) cannot be coherent
- Percentage of references to memory for writing in the order of 15% (some studies).



Example of cache in other systems





Memory Systems
Cache, DRAM, Disk
Bruce Jacob, Spencer Ng, David Wang
Elsevier

ARCOS Group

uc3m | Universidad Carlos III de Madrid

Lesson 5 (II)
Memory hierarchy

Computer Structure

Bachelor in Computer Science and Engineering