

Grupo ARCOS

**uc3m** | Universidad **Carlos III** de Madrid

## Tema 5 (II)

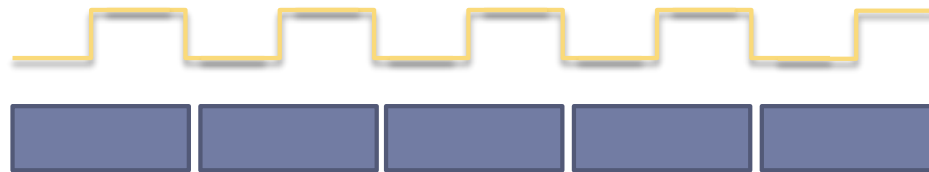
### Jerarquía de Memoria

# Contenidos

1. Tipos de memoria
2. Jerarquía de memoria
3. Memoria principal
4. Memoria caché

# Característica de la memoria principal

- ▶ Se premia el acceso a posiciones consecutivas de memoria
- ▶ Ejemplo 1: acceder a 5 posiciones de memoria **individuales**



- ▶ Ejemplo 2: acceder a 5 posiciones de memoria **consecutivas**



# Característica de los accesos a memoria

- ▶ “Principio de proximidad o localidad de referencias”:

Durante la ejecución de un programa,  
las referencias (direcciones) a memoria tienden a estar  
agrupadas por:

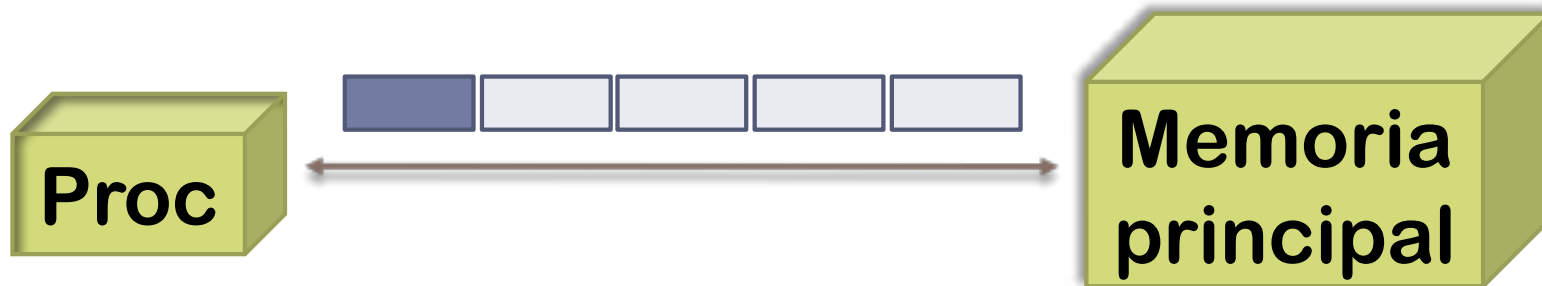
- ▶ **proximidad espacial**
  - ▶ Secuencia de instrucciones
  - ▶ Acceso secuencial a arrays
- ▶ **proximidad temporal**
  - ▶ Bucles

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

.text
main: li $t0 0
      la $t1 vector
      b2: bge $t0 1024 finb2
          mult $t2 $t0 4
          add $t2 $t1 $t2
          sw $t0 ($t2)
          add $t0 $t0 1
          b b2
      finb2: jr $ra
```

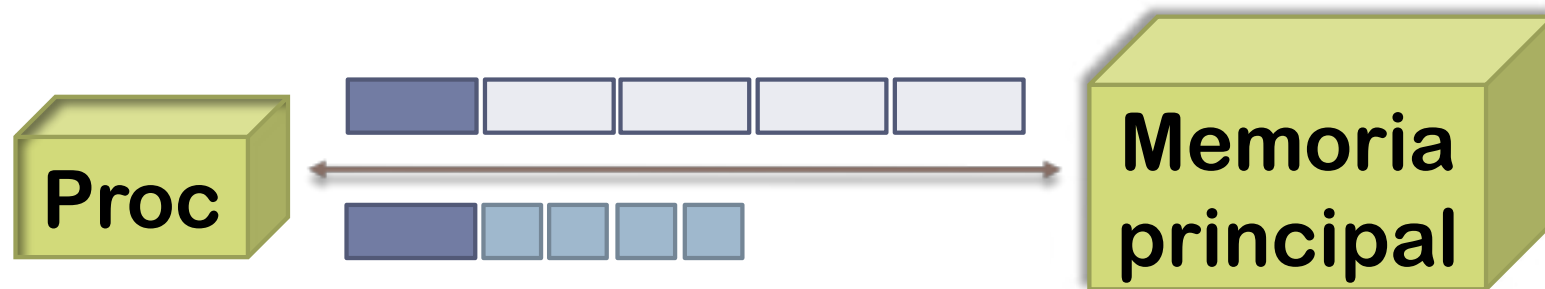
# Objetivo de la memoria caché: aprovechar los accesos contiguos

- ▶ Si cuando se accede a una posición de memoria solo se transfieren los datos de esa posición, no se aprovecha los posibles accesos a datos contiguos.



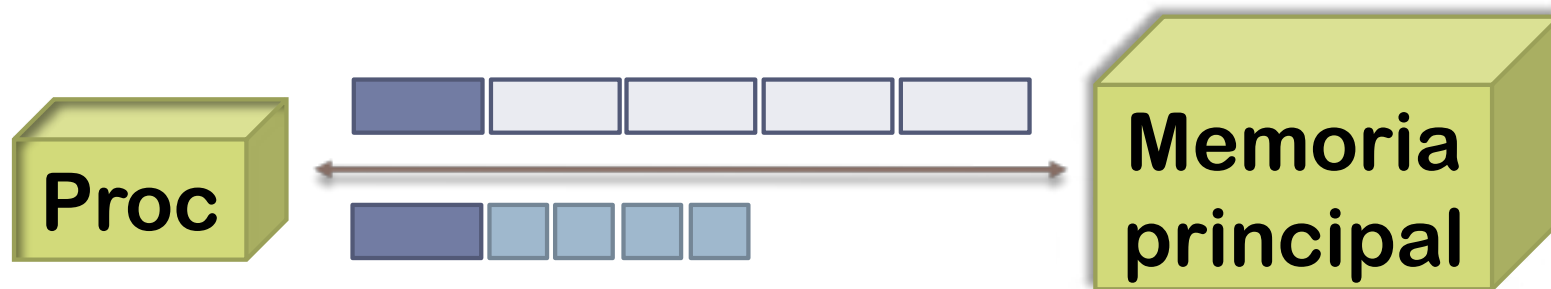
# Objetivo de la memoria caché: aprovechar los accesos contiguos

- ▶ Si cuando se accede a una posición de memoria se transfiere esos datos y los contiguos, sí se aprovecha el acceso a datos contiguos



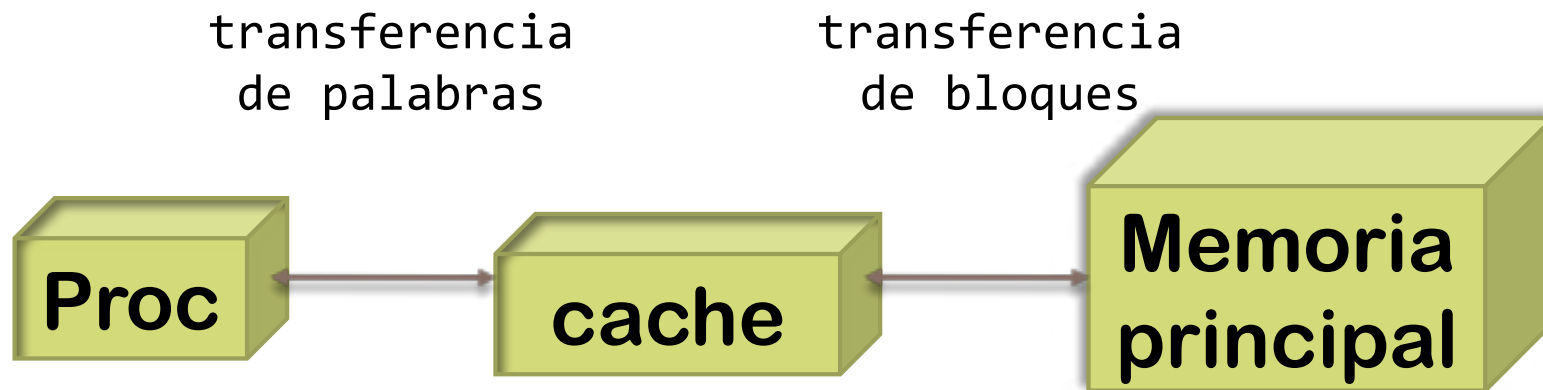
# Objetivo de la memoria caché: aprovechar los accesos contiguos

- ▶ Si cuando se accede a una posición de memoria se transfiere esos datos y los contiguos, sí se aprovecha el acceso a datos contiguos
  - ▶ Transfiero de la memoria principal un bloque de palabras
  - ▶ ¿Dónde se almacenan las palabras del bloque?



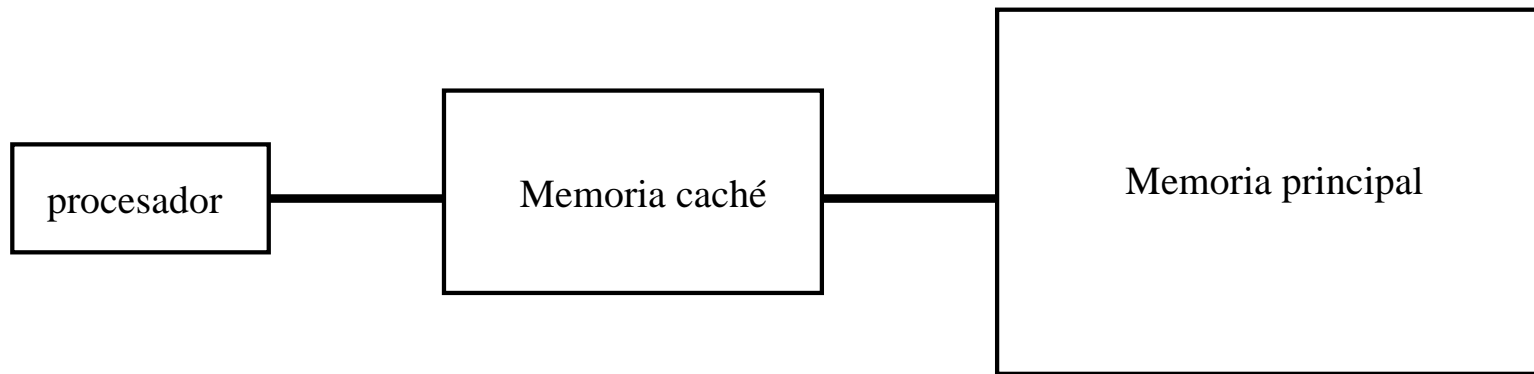
# Memoria Cache

- ▶ Cantidad pequeña de memoria rápida SRAM
  - ▶ Integrada en el mismo procesador
  - ▶ Más rápida y cara que la memoria principal DRAM
- ▶ Está entre la memoria principal y el procesador
- ▶ Almacena una **copia** de partes de la memoria principal

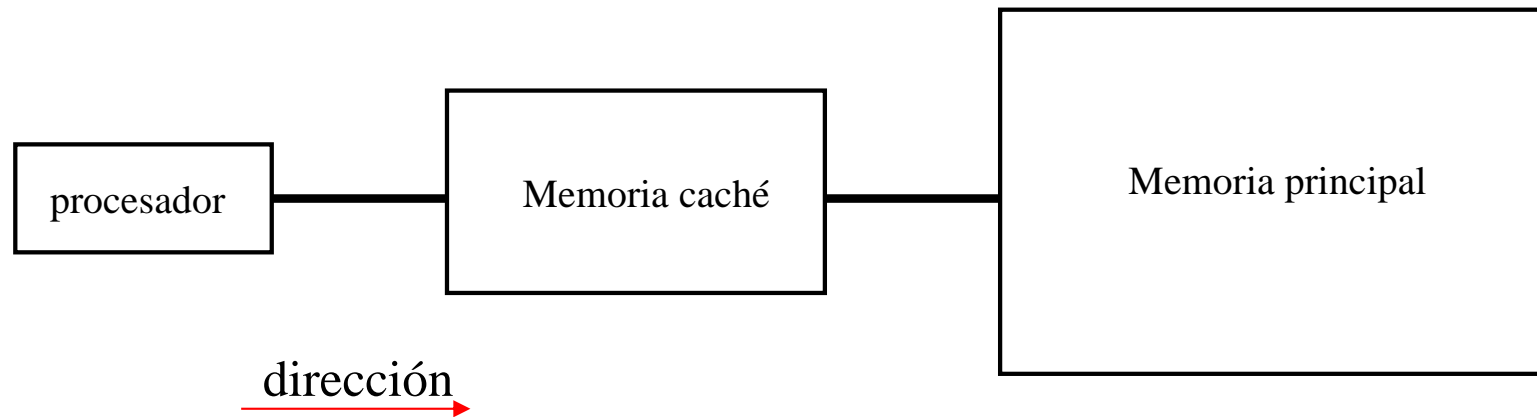




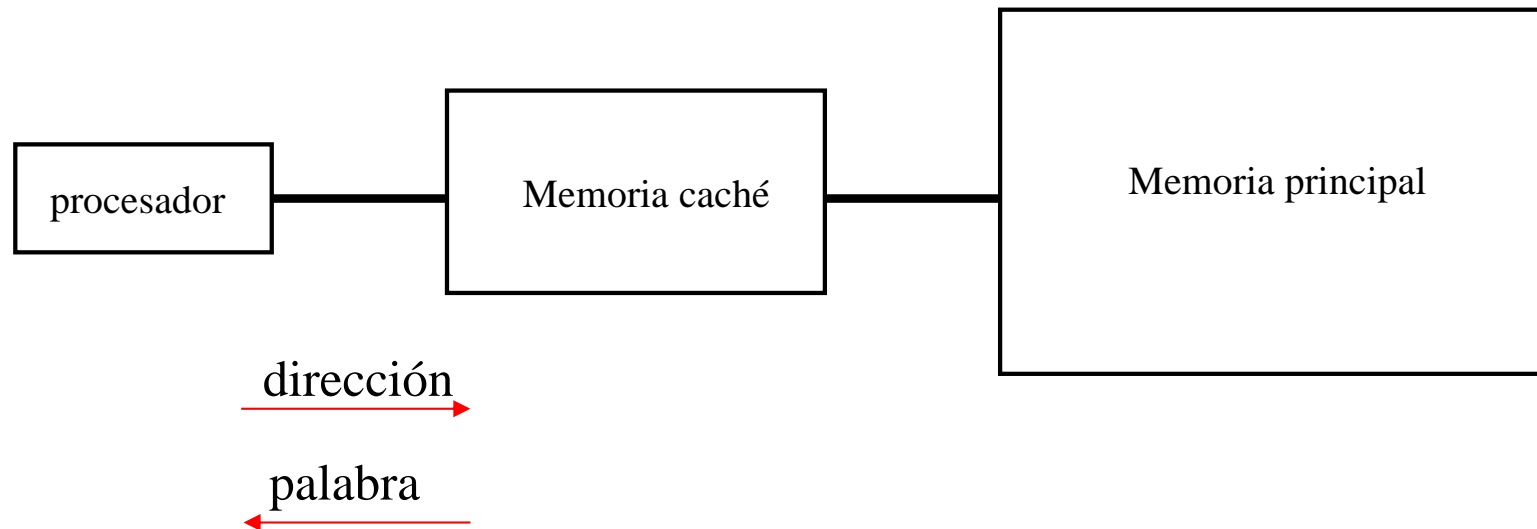
# Funcionamiento de la memoria caché



# Funcionamiento de la memoria caché

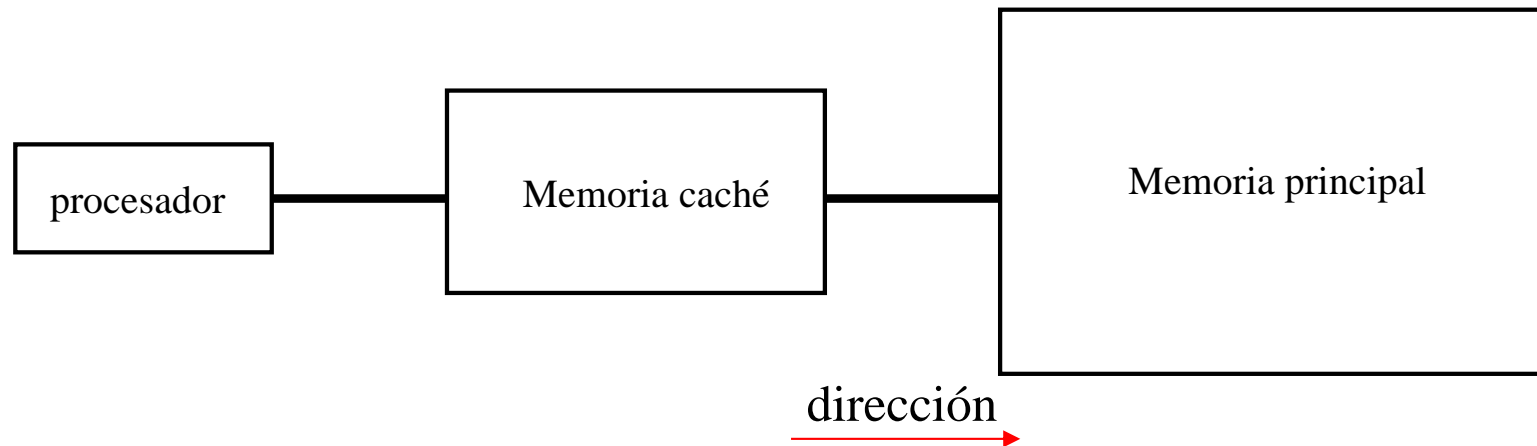


# Funcionamiento de la memoria caché



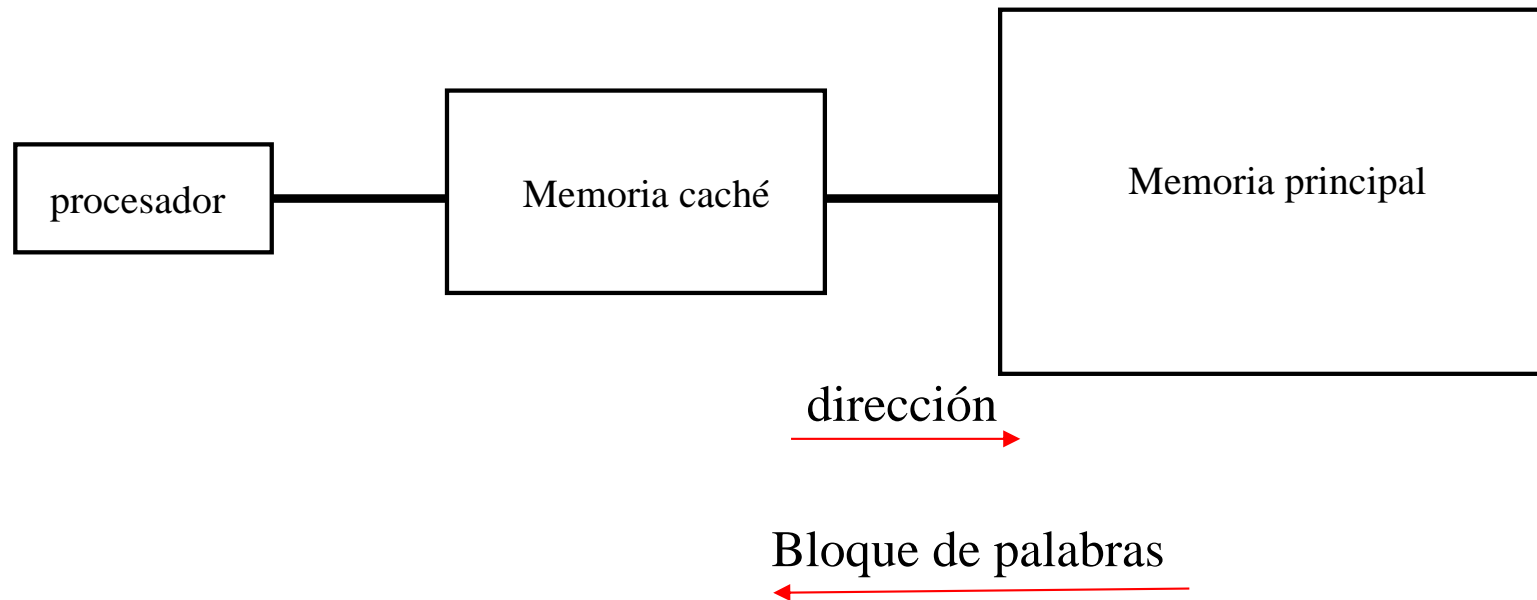
ACIERTO

# Funcionamiento de la memoria caché

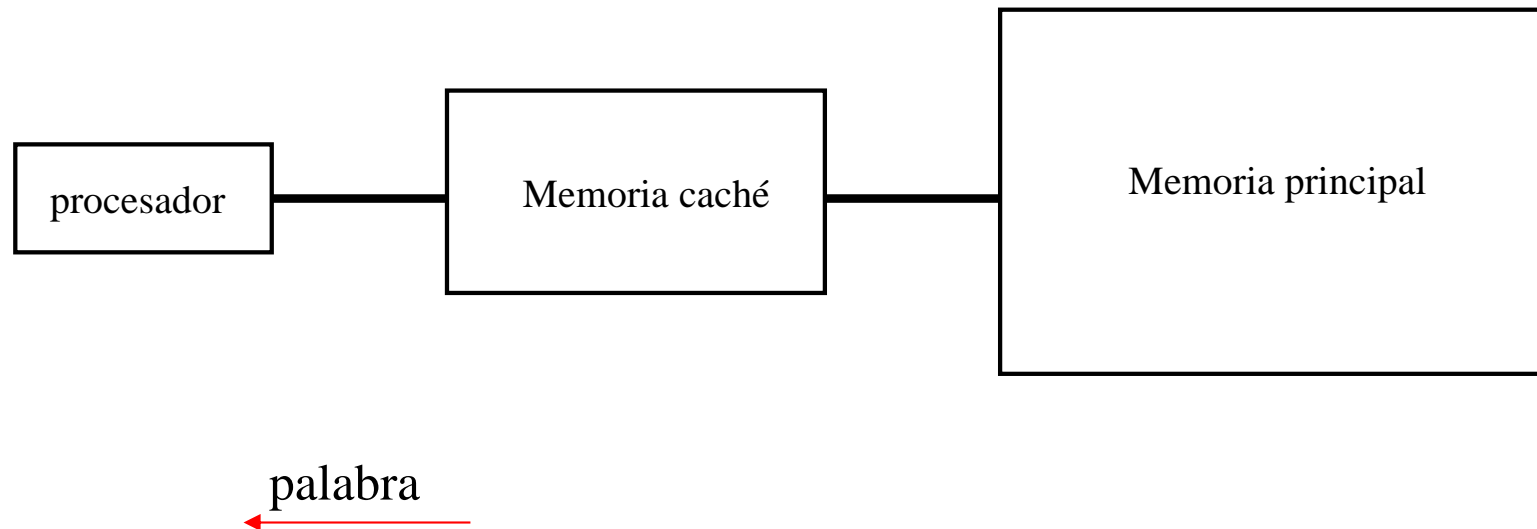


FALLO

# Funcionamiento de la memoria caché

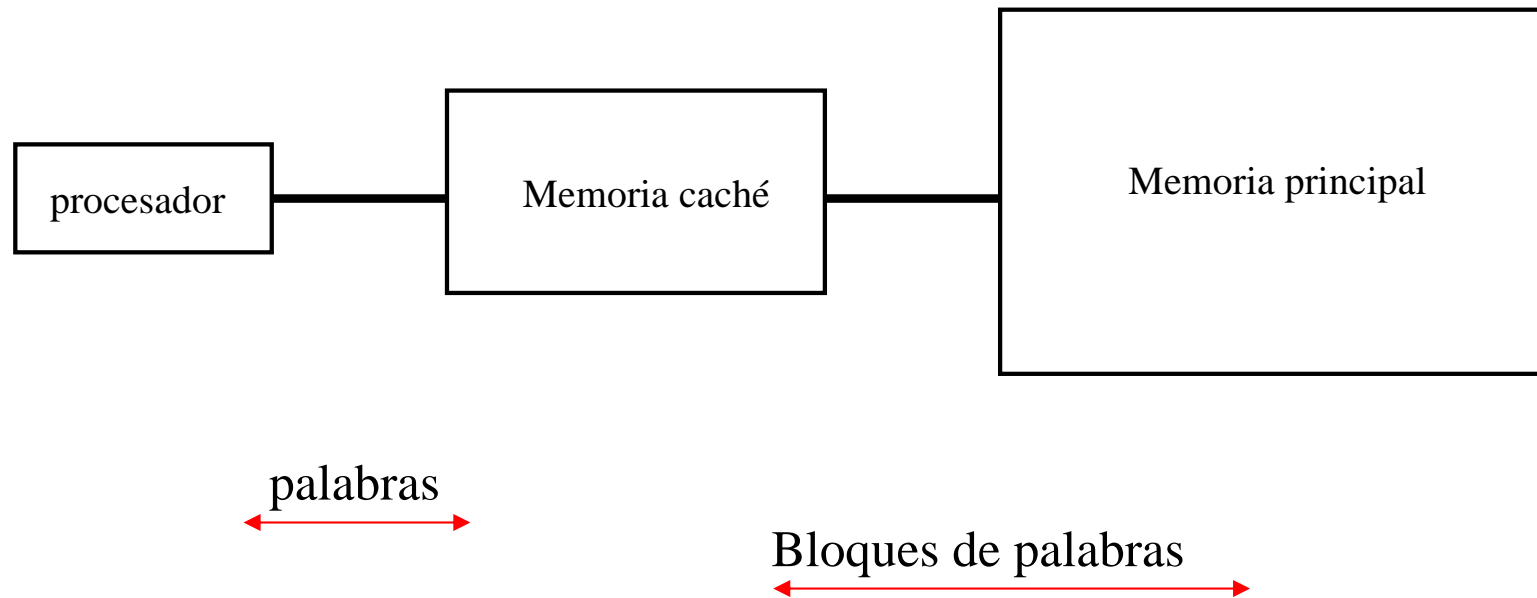


# Funcionamiento de la memoria caché

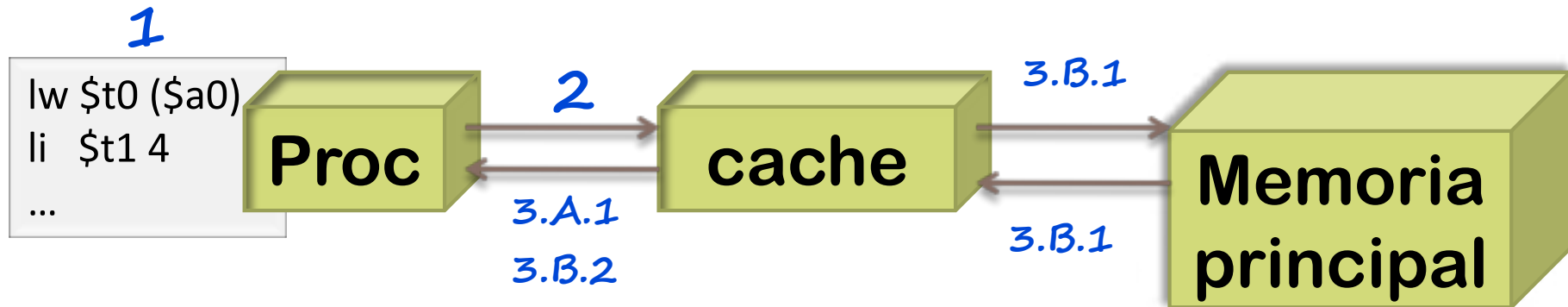


# Funcionamiento de la memoria caché

## Resumen



# Funcionamiento general



1. El procesador solicita contenidos de una posición de memoria.
2. La cache comprueba si ya están los datos de esta posición:
  - ▶ **Si está (ACIERTO)**
    - 3.A.1** Se la sirve al procesador desde la cache (rápidamente).
  - ▶ **Si no está (FALLO),**
    - 3.B.1** La cache transfiere de memoria principal el bloque asociado a la posición.
    - 3.B.2** Después, la cache entrega los datos pedidos al procesador



# Ejemplo de funcioamiento

```
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  
b     bucle  
fin:  ...
```

- ▶ Ejemplo:
  - ▶ Acceso a caché: 2 ns
  - ▶ Acceso a MP: 120 ns
  - ▶ Bloque de caché: 4 palabras
  - ▶ Transferencia de un bloque entre memoria principal y caché: 200 ns

# Ejemplo de funcionamiento

```
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
      b      bucle
fin:   ...
```

► Sin memoria caché:

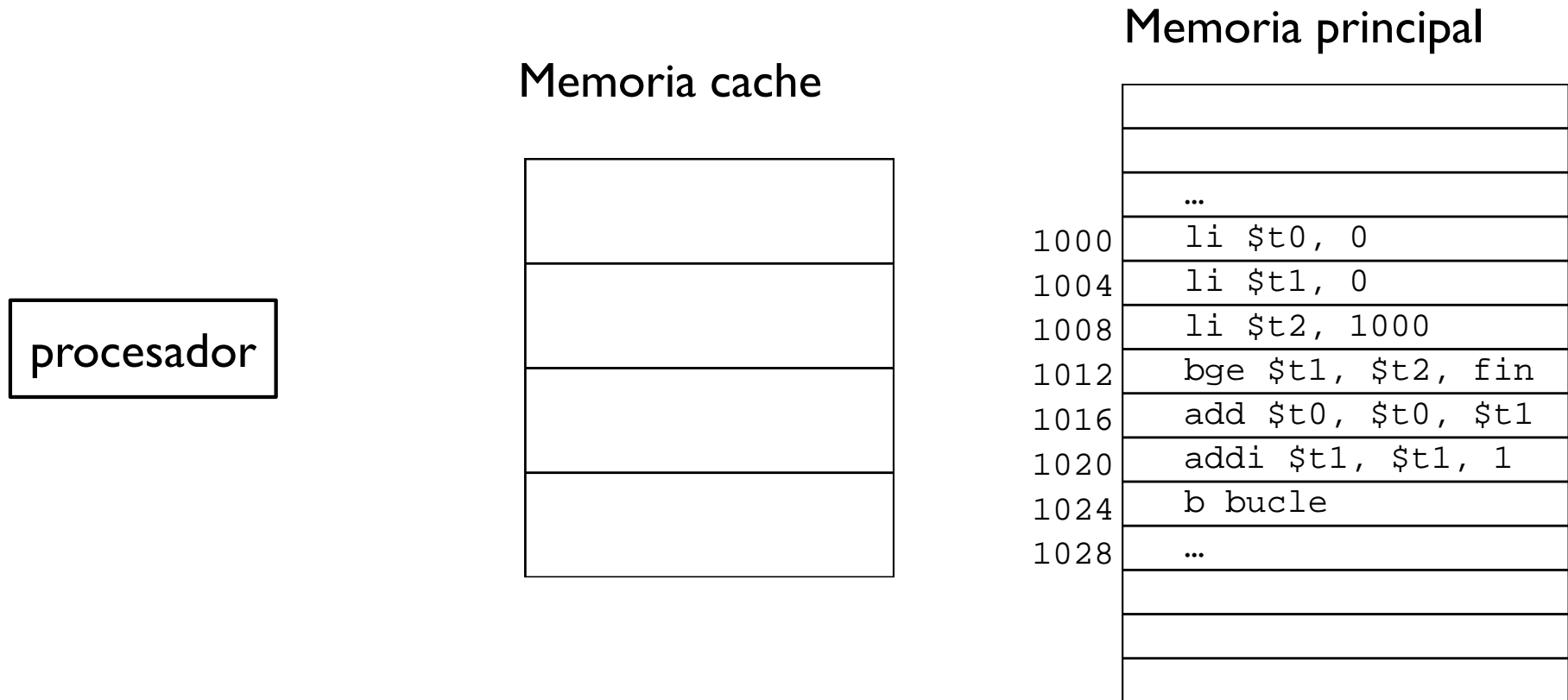
- Número de accesos a memoria =  $3 + 4 \times 1000 + 1 = 4004$  accesos
- Tiempo de acceso a memoria =  $4004 \times 120 = 480480$  ns = 480.480 ms

# Ejemplo de funcionamiento

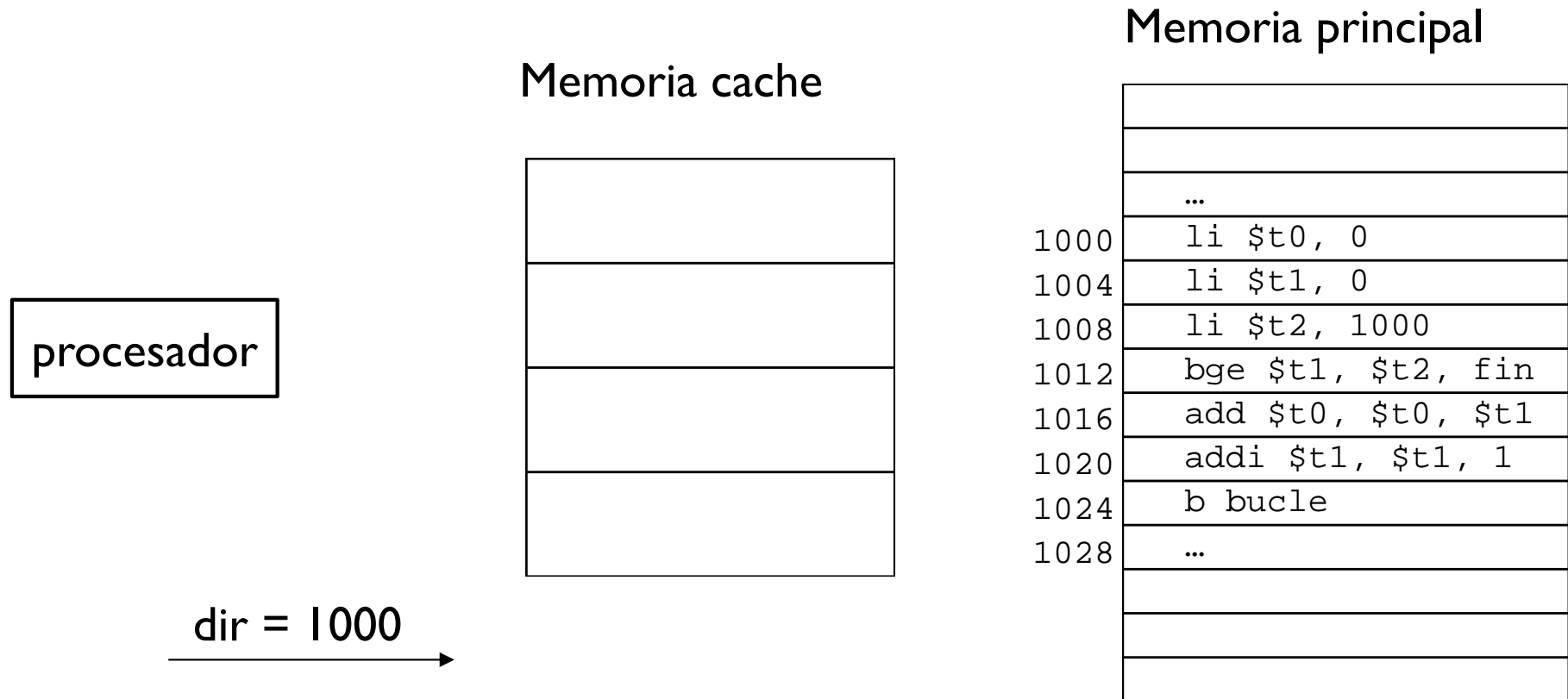
```
int i;                                li    $t0, 0    // s
int s = 0;                            li    $t1, 0    // i
for (i=0; i < 1000; i++)              li    $t2, 1000
    s = s + i;                        bucle: bge    $t1, $t2, fin
                                      add    $t0, $t0, $t1
                                      addi   $t1, $t1, 1
                                      b      bucle
                                      fin:   ...
```

- Con memoria caché (bloque de 4 palabras):

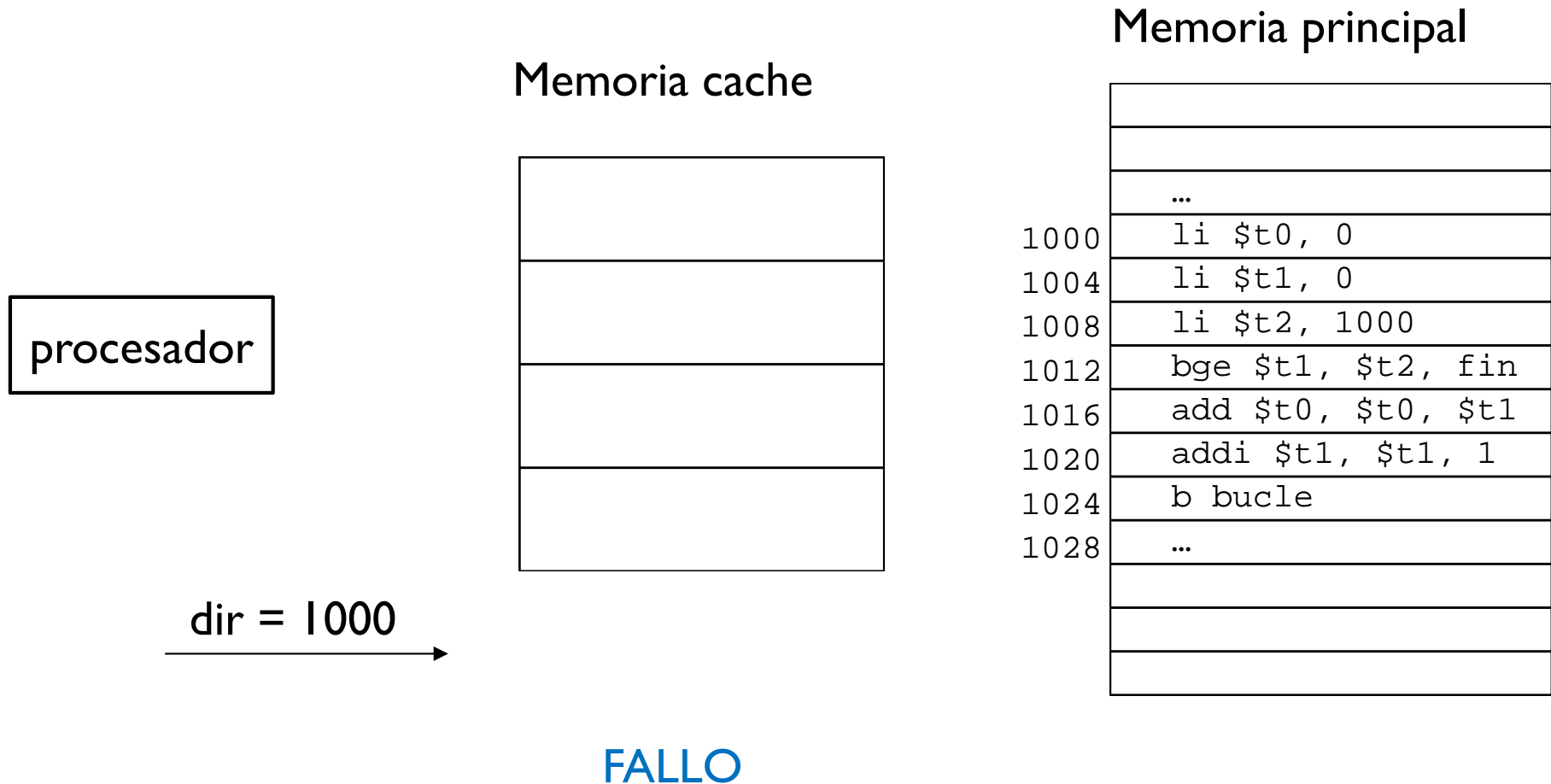
# Ejemplo de funcionamiento



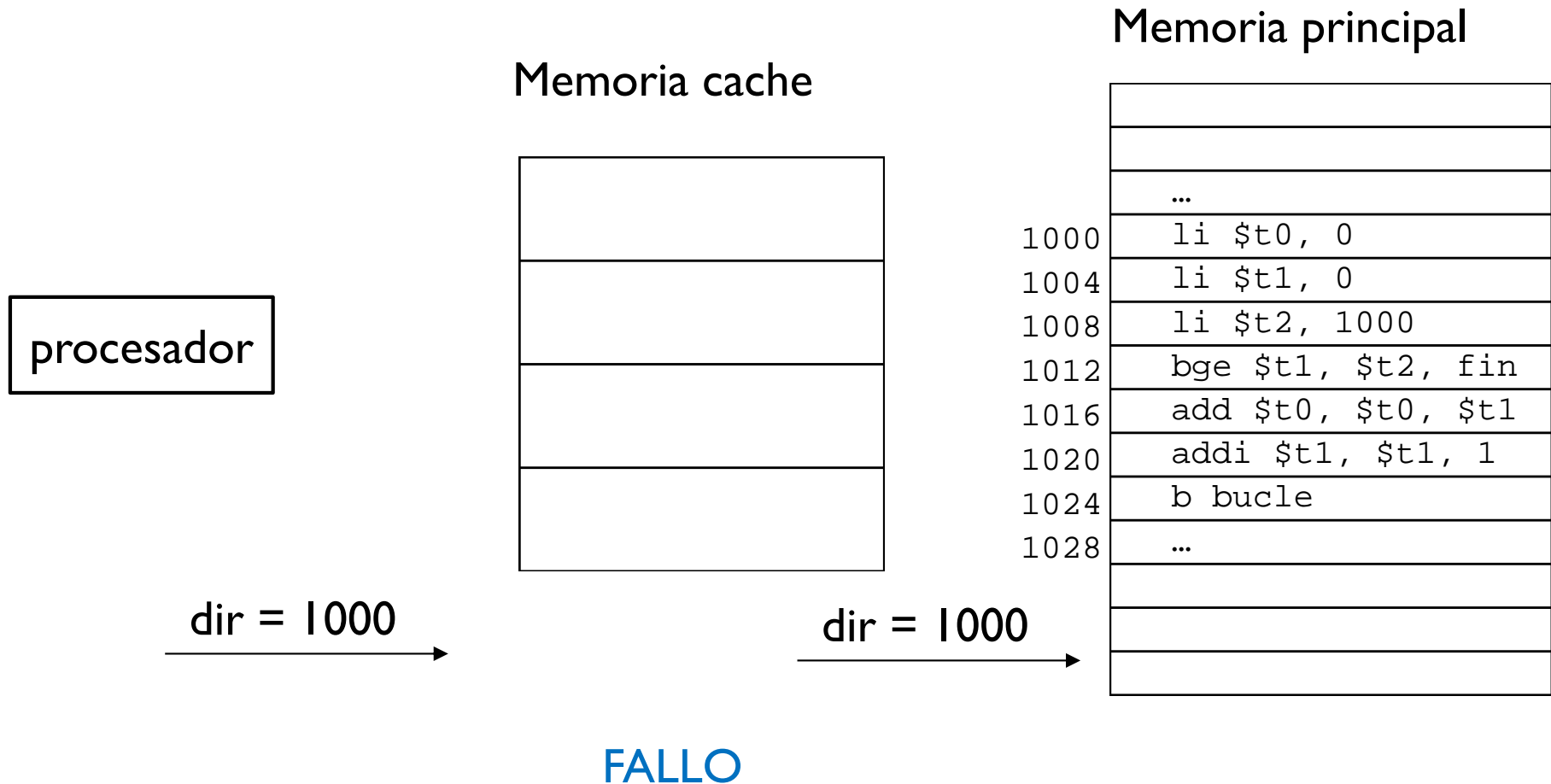
# Ejemplo de funcionamiento



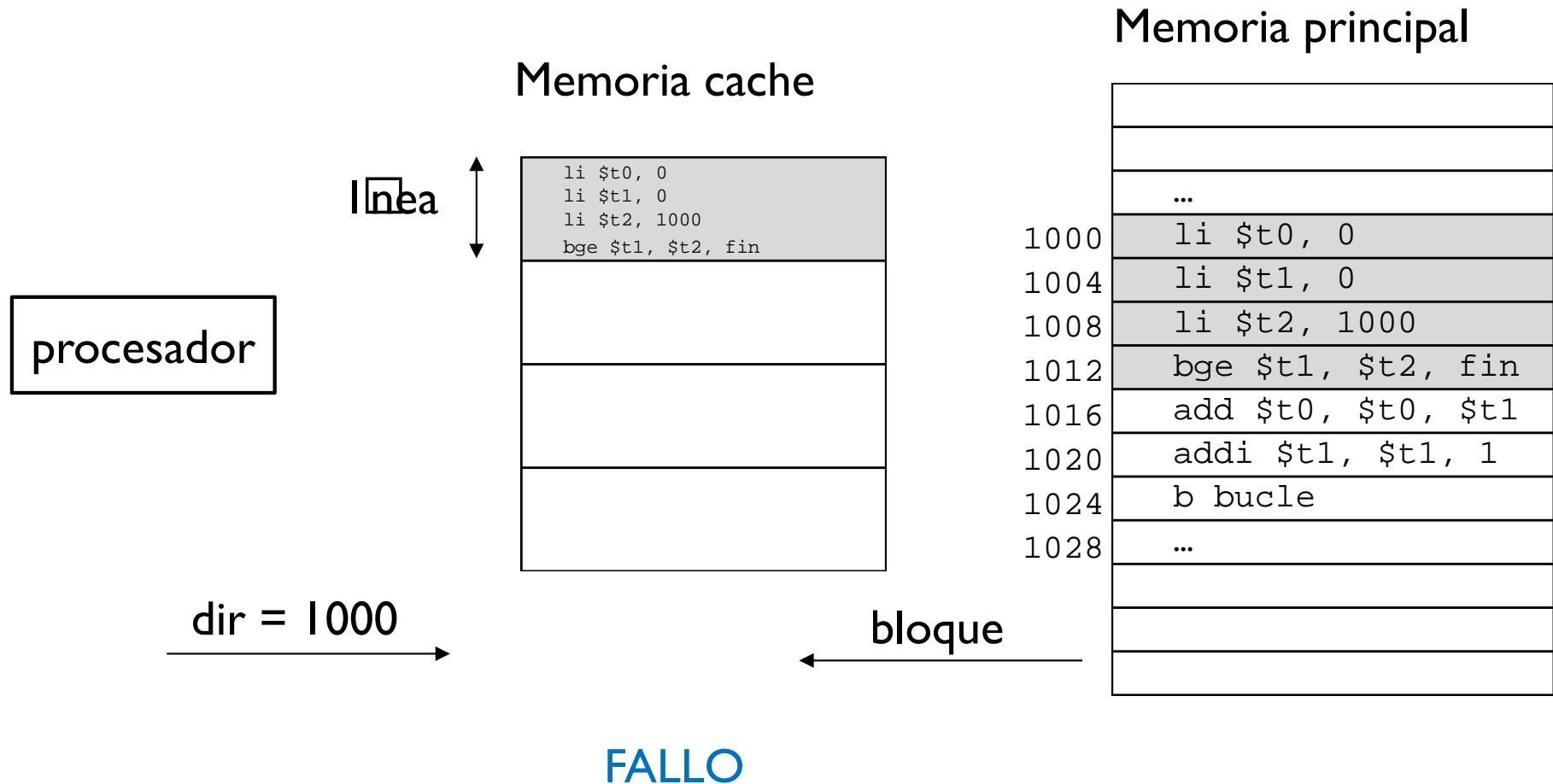
# Ejemplo de funcionamiento



# Ejemplo de funcionamiento

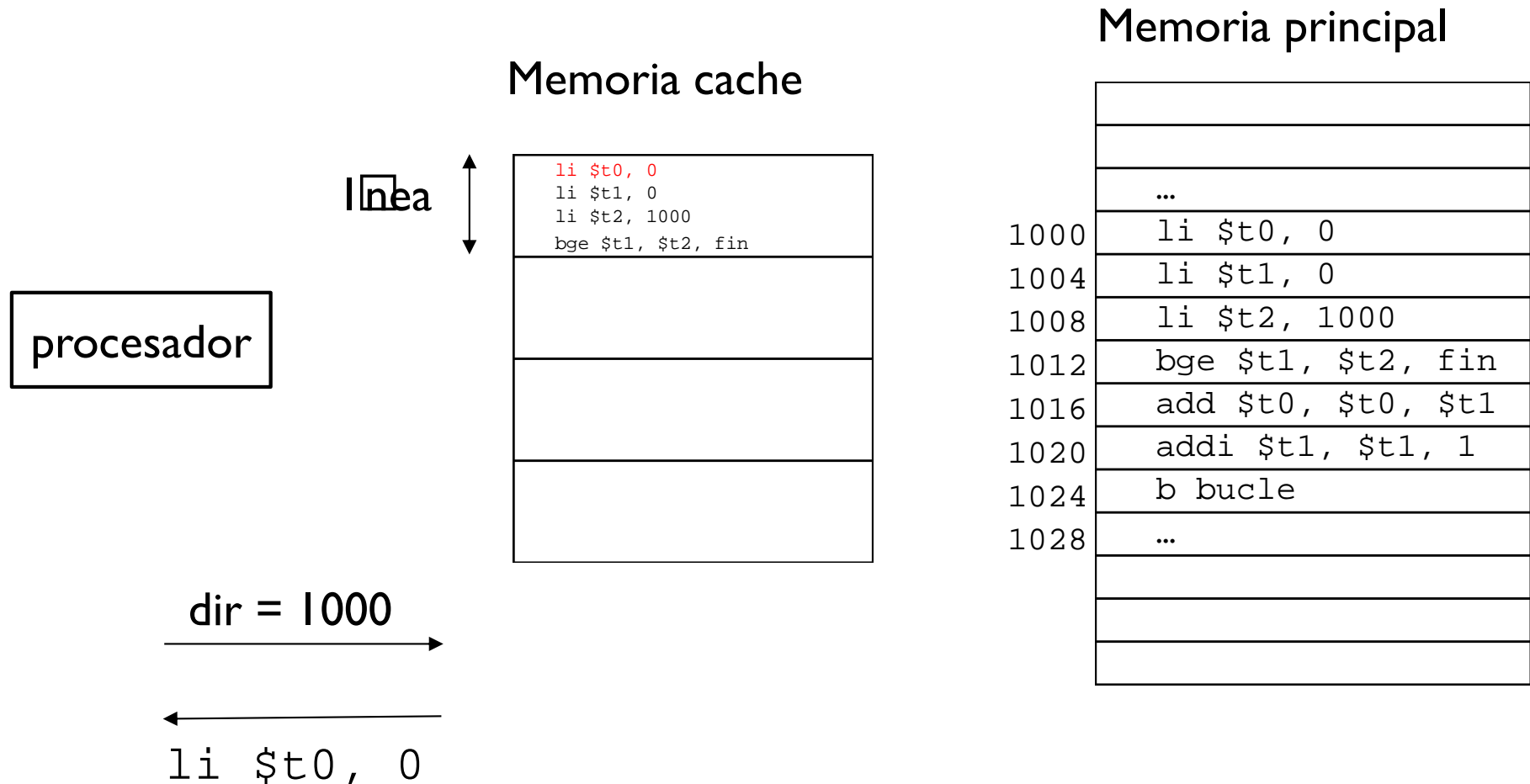


# Ejemplo de funcionamiento

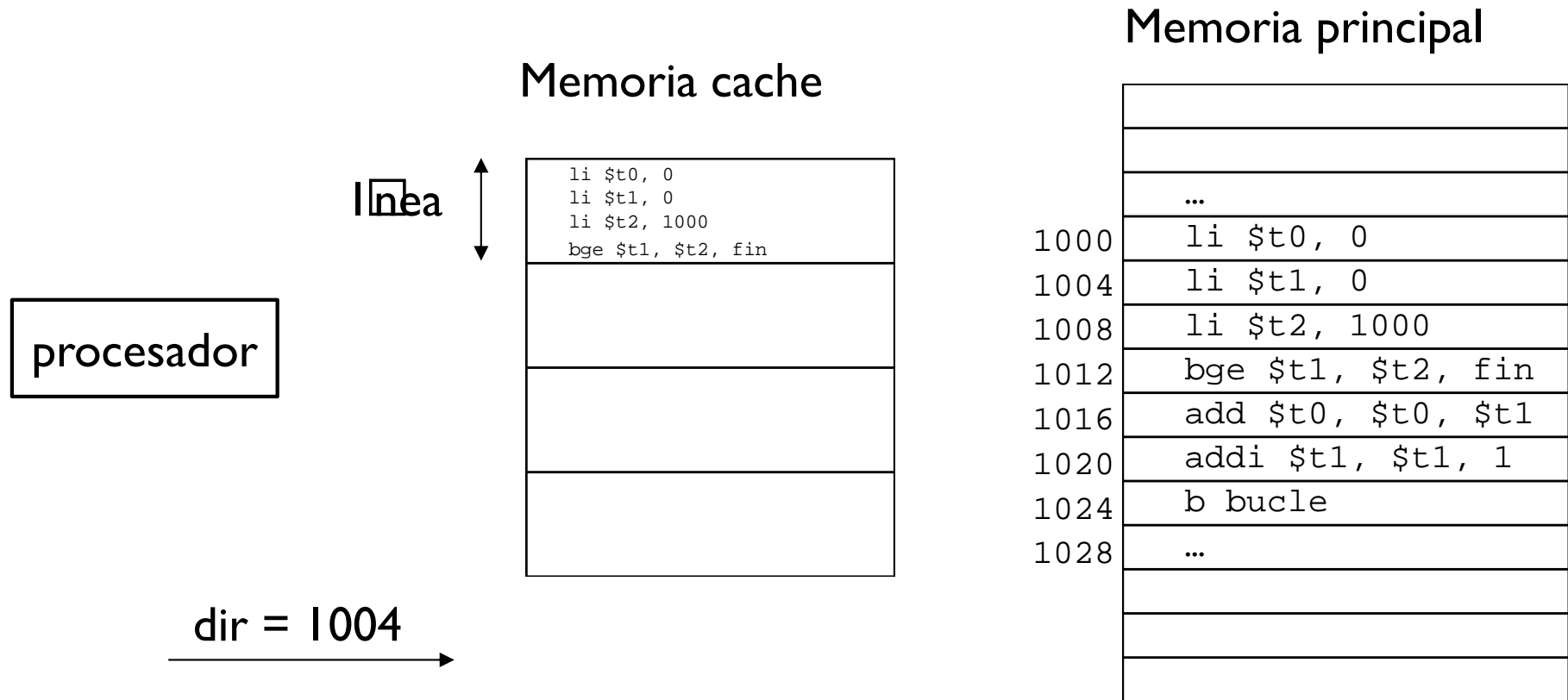




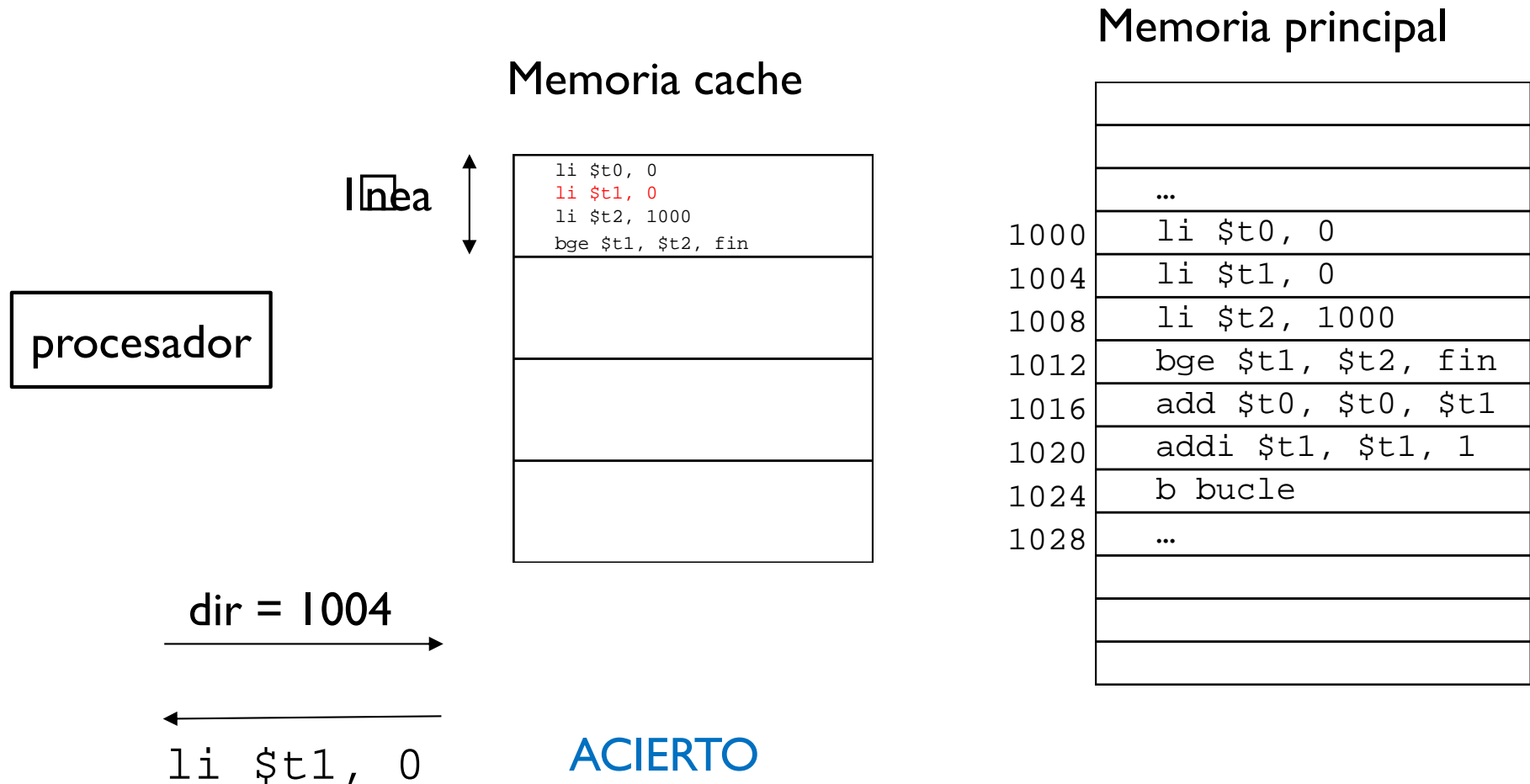
# Ejemplo de funcionamiento



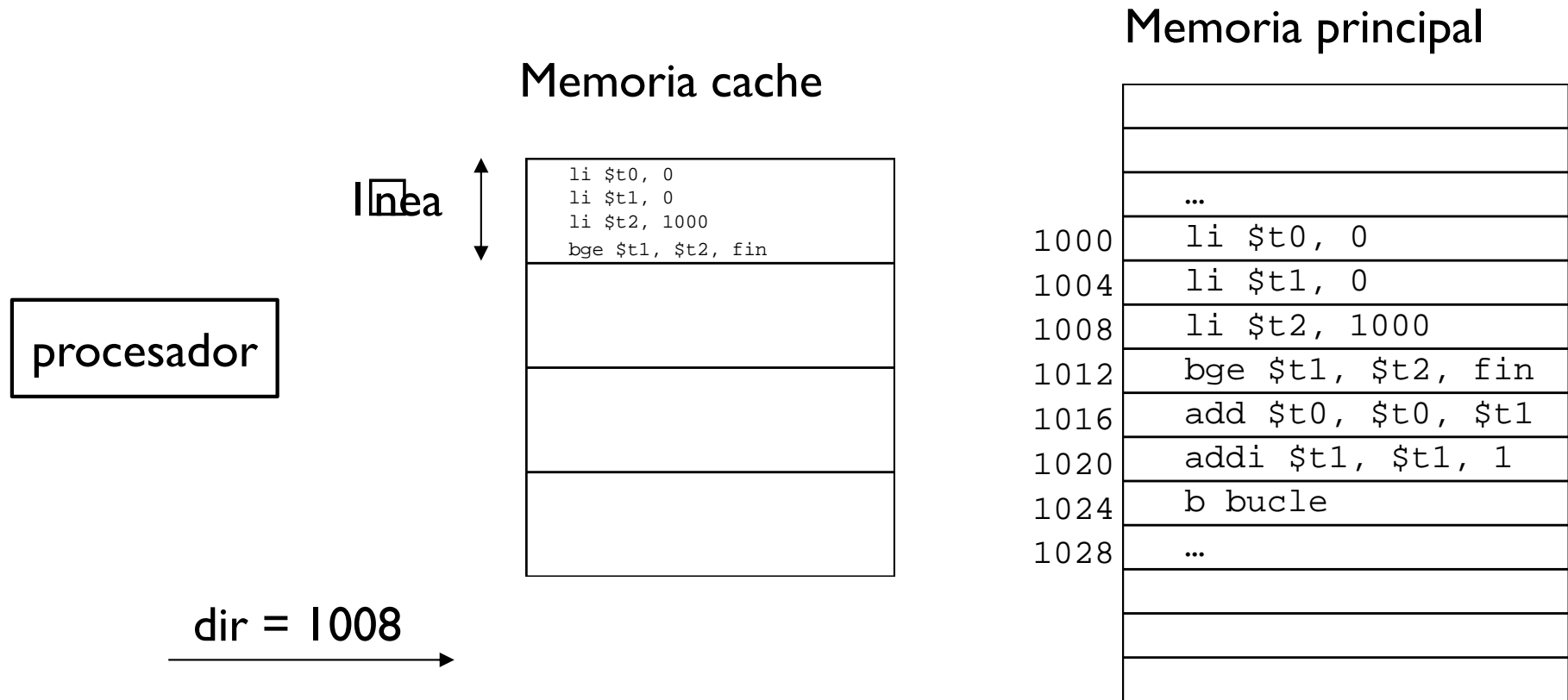
# Ejemplo de funcionamiento



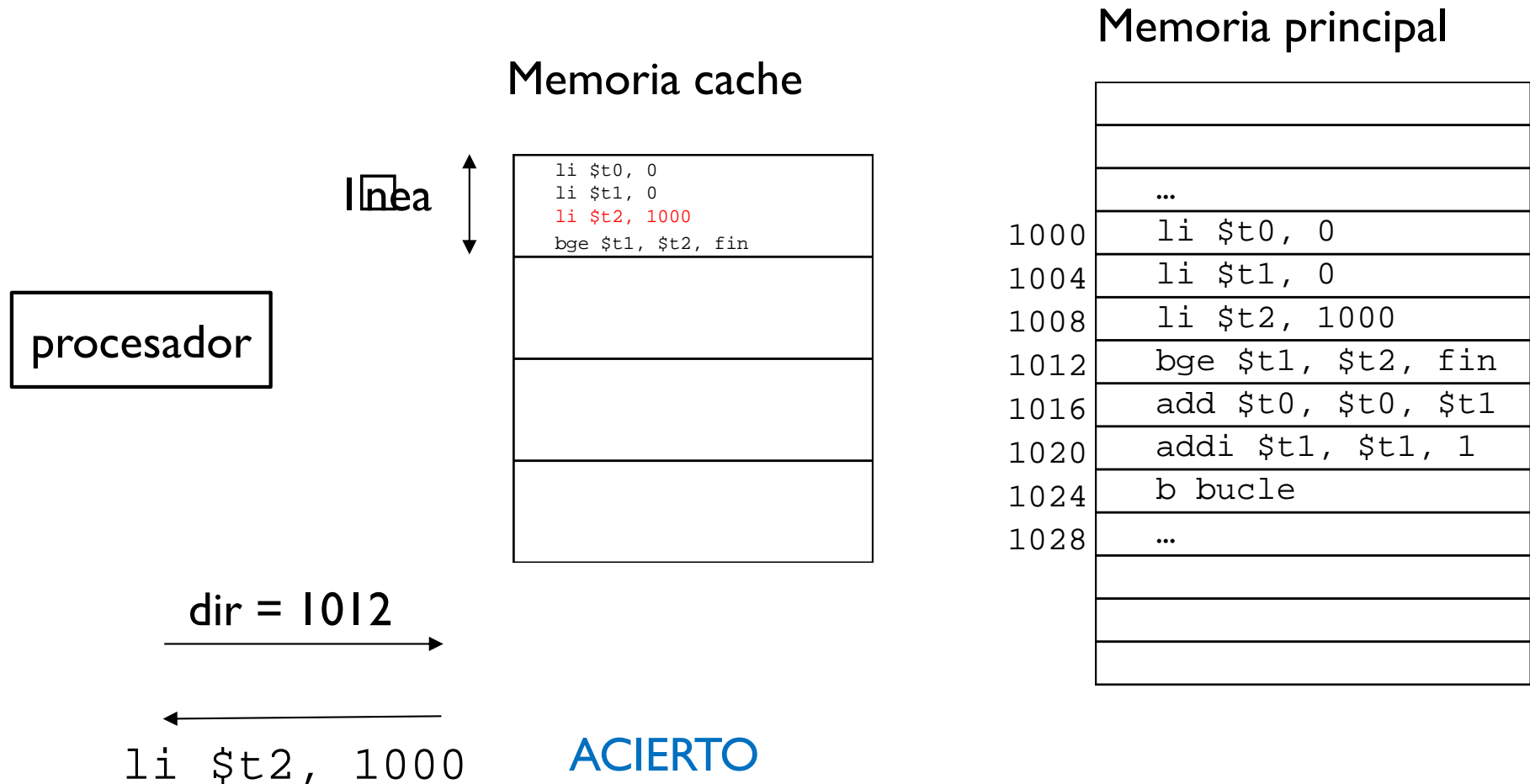
# Ejemplo de funcionamiento



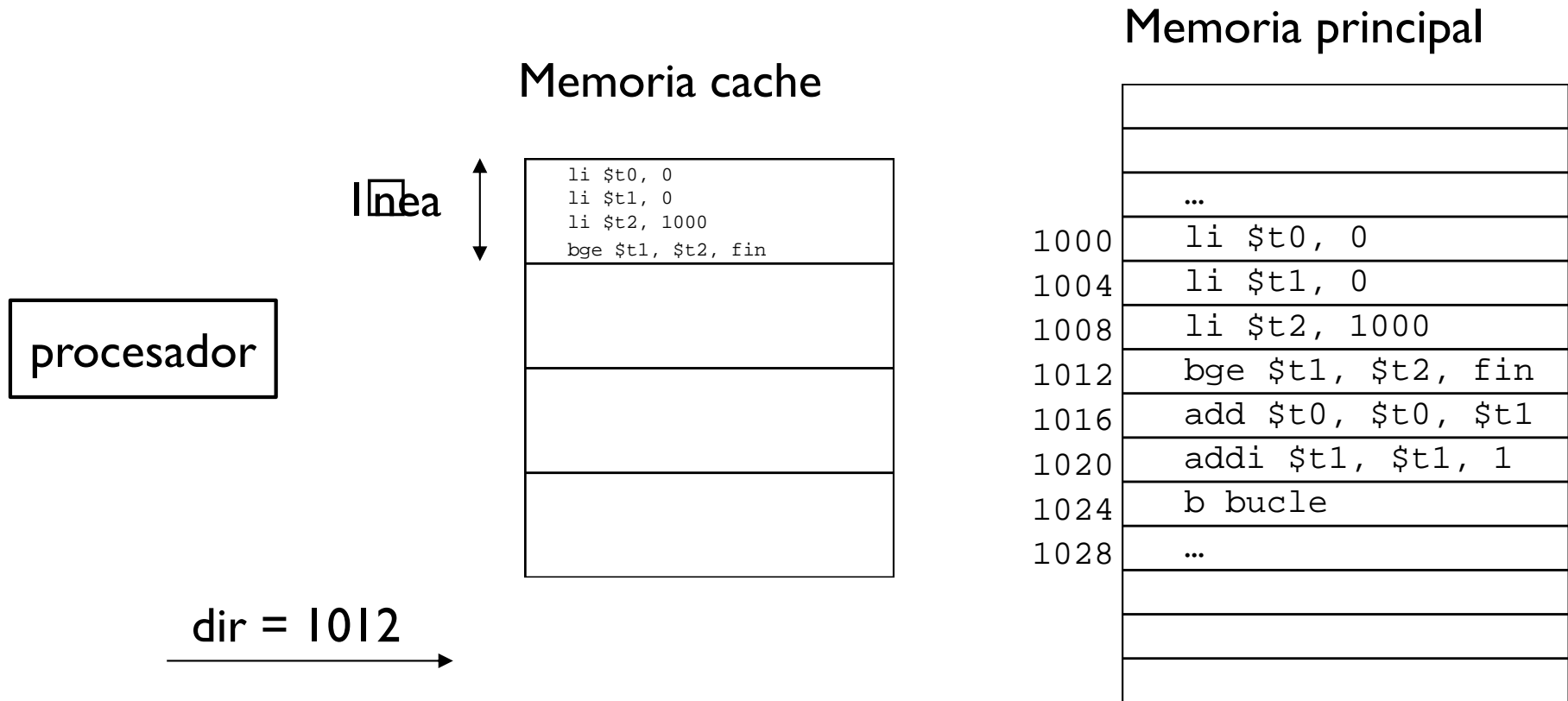
# Ejemplo de funcionamiento



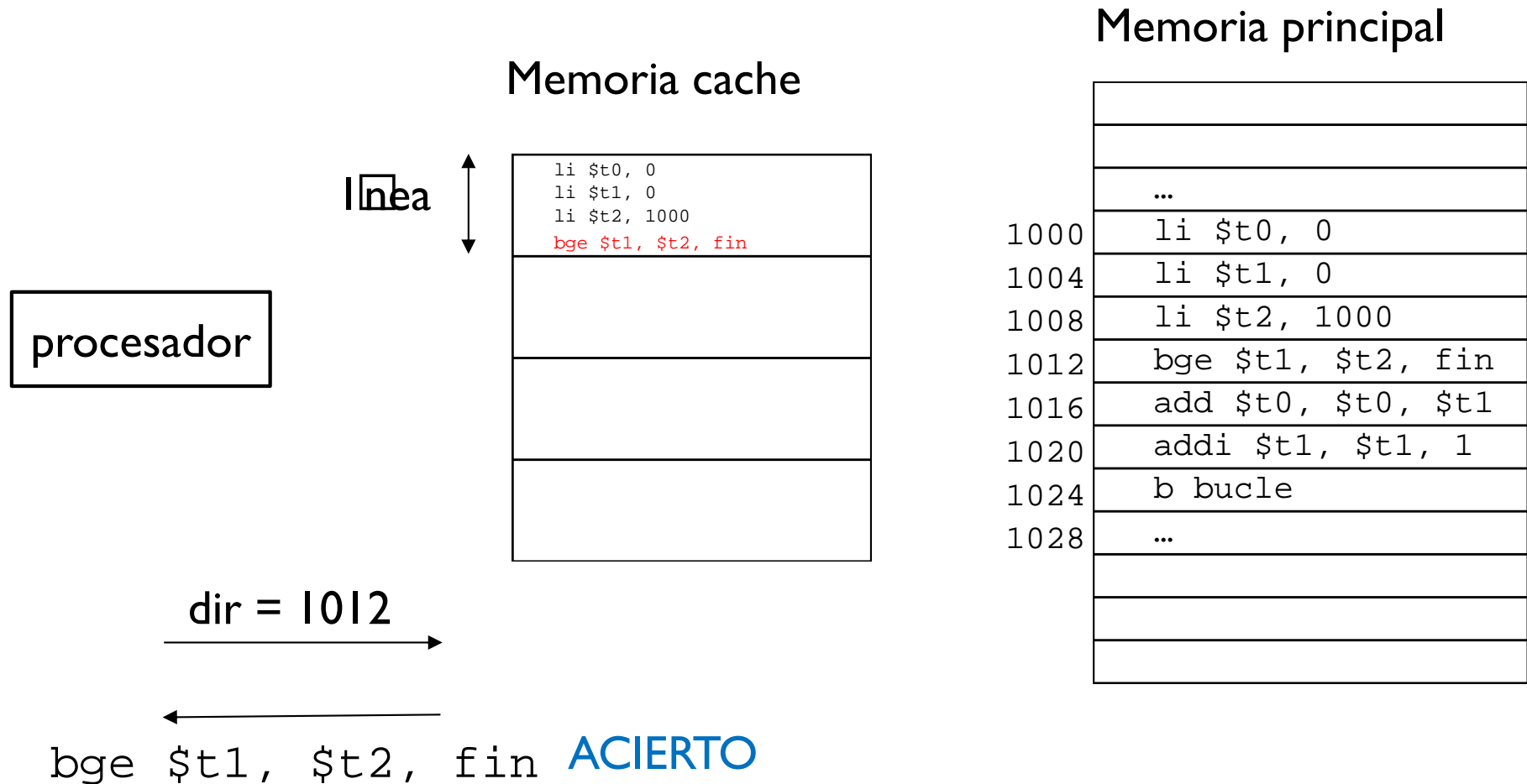
# Ejemplo de funcionamiento



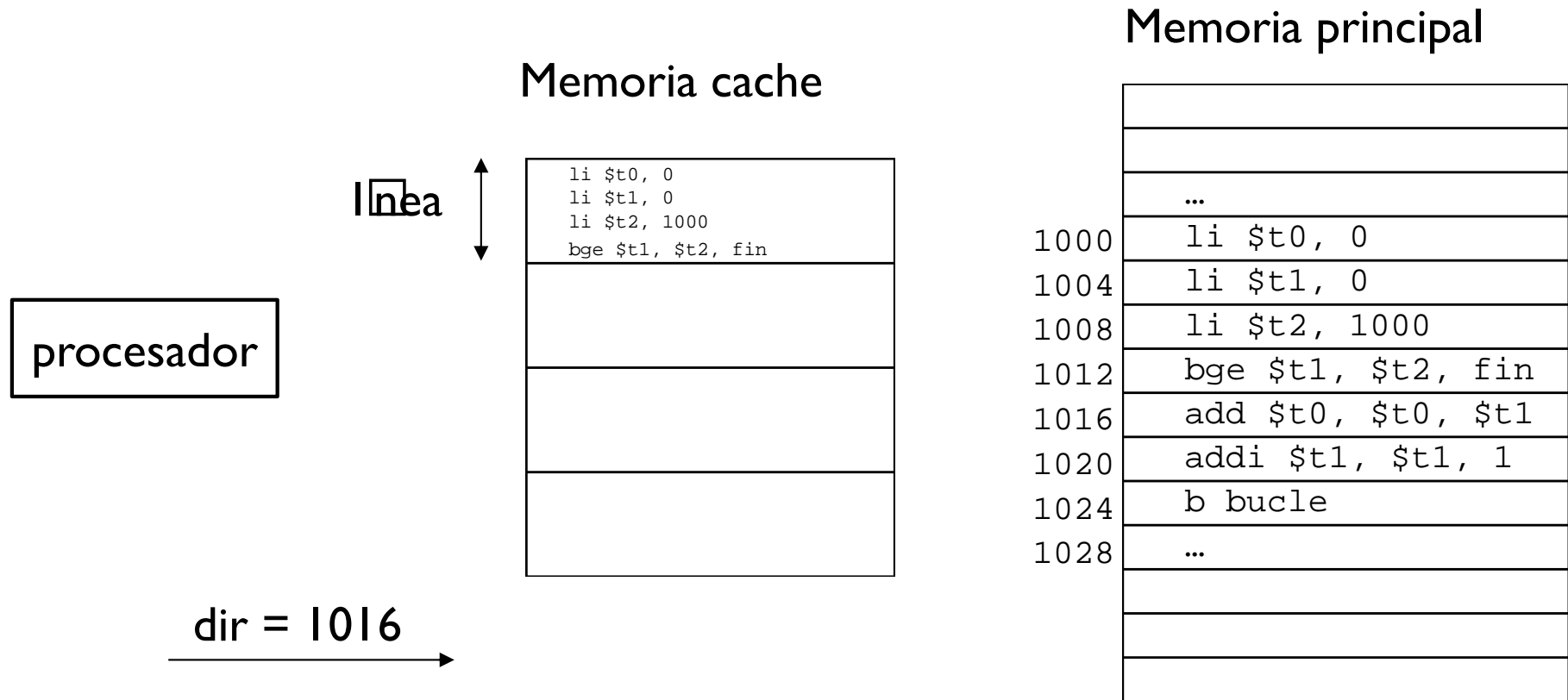
# Ejemplo de funcionamiento



# Ejemplo de funcionamiento

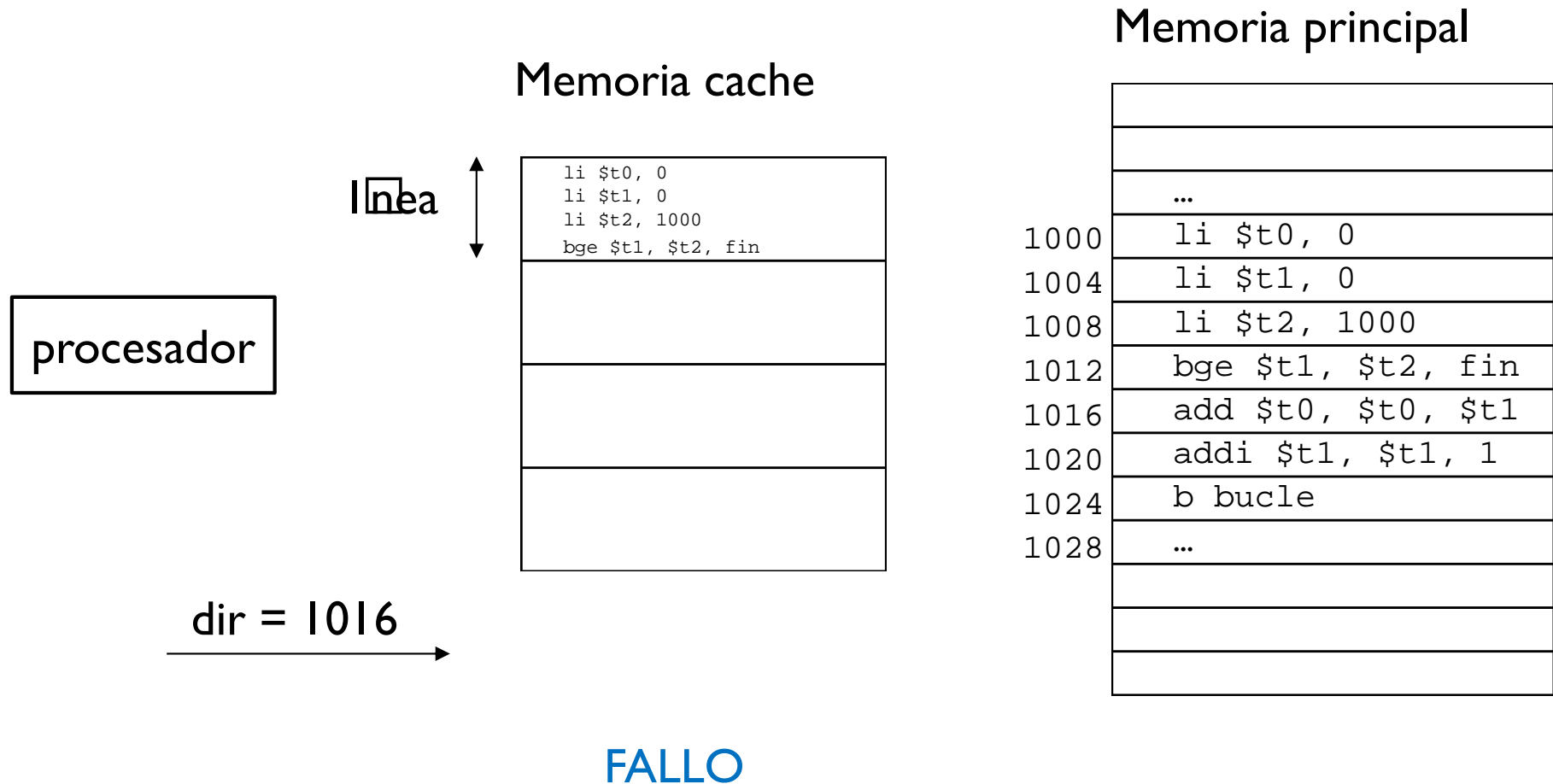


# Ejemplo de funcionamiento

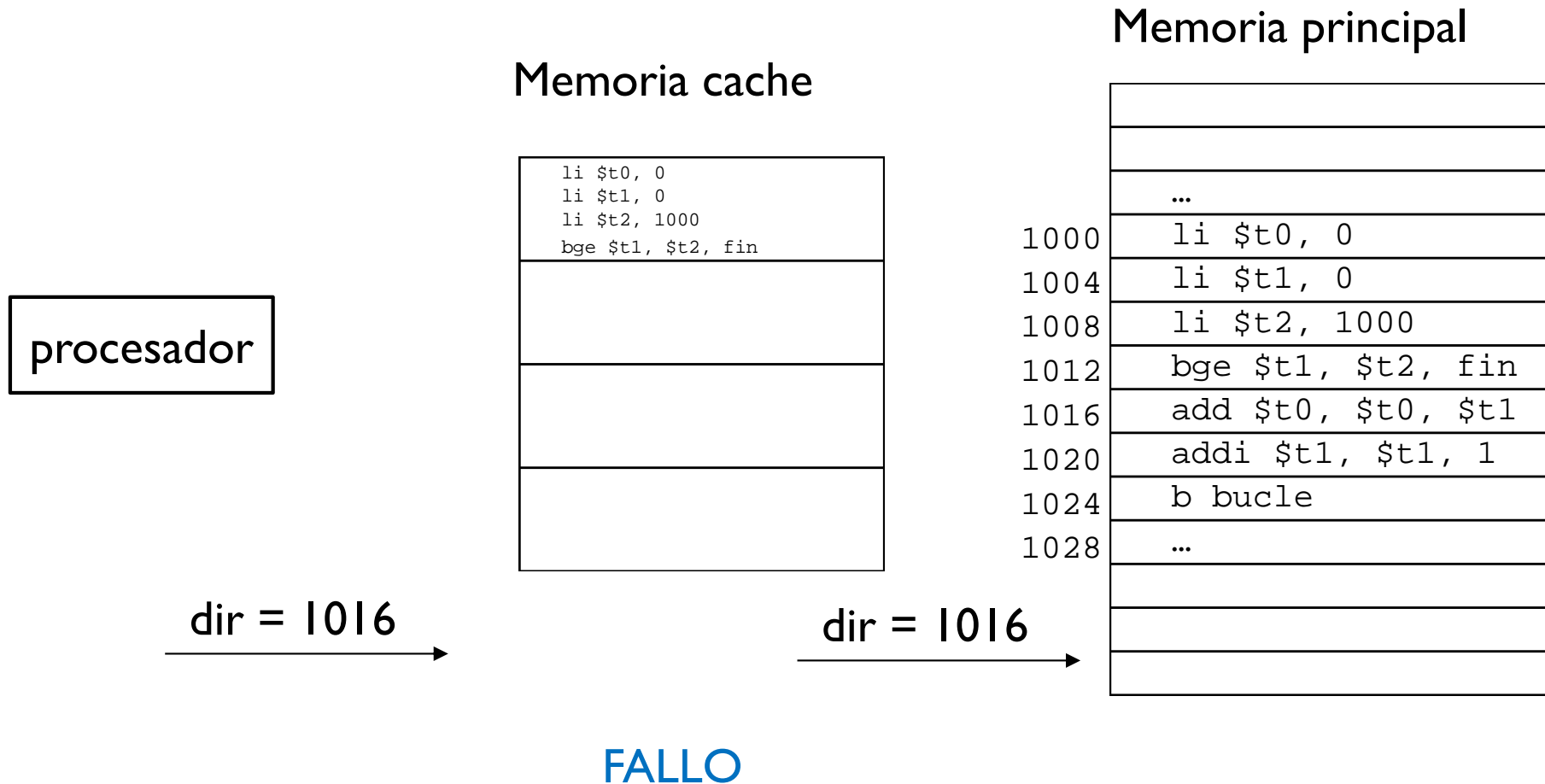




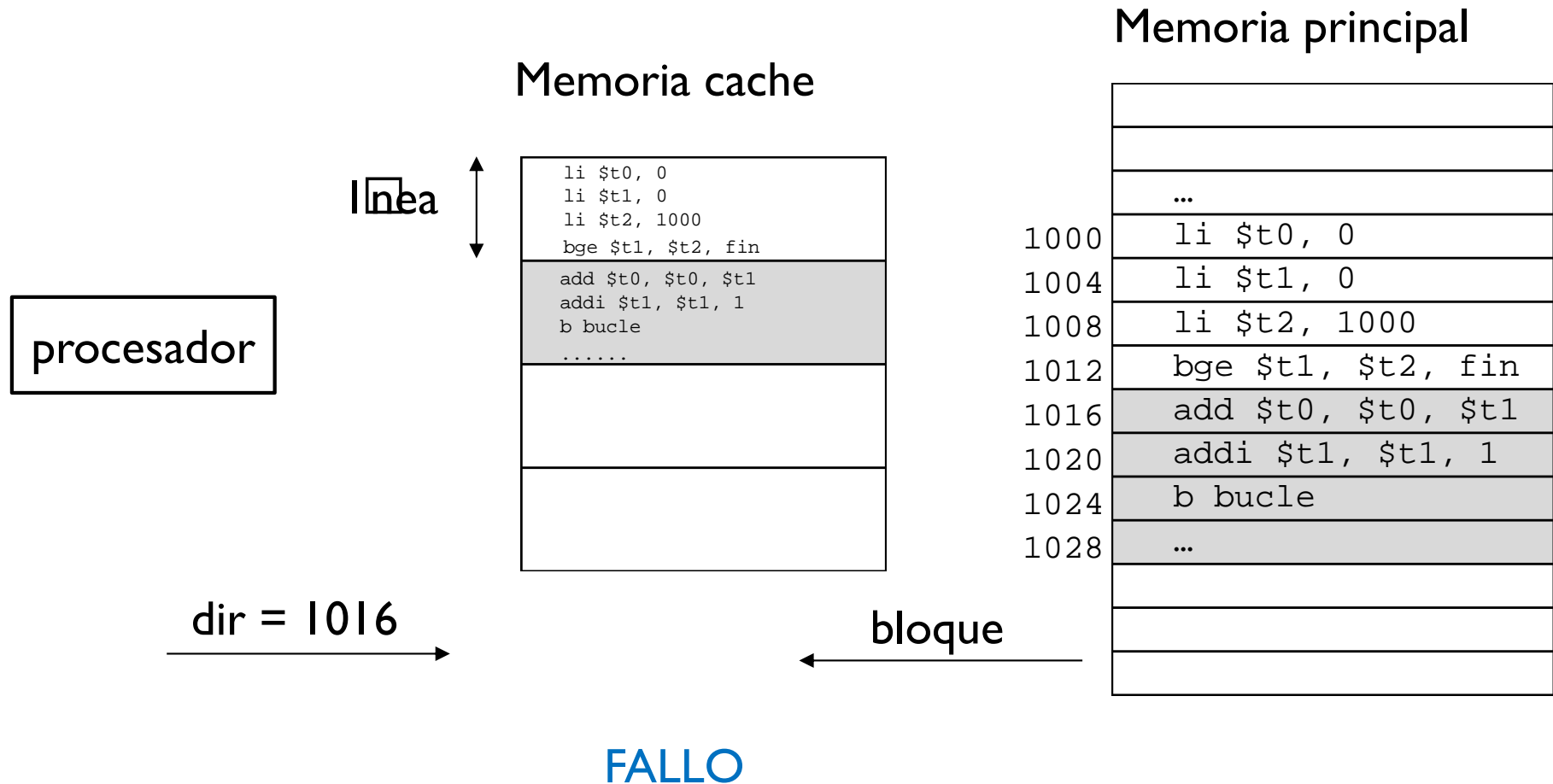
# Ejemplo de funcionamiento



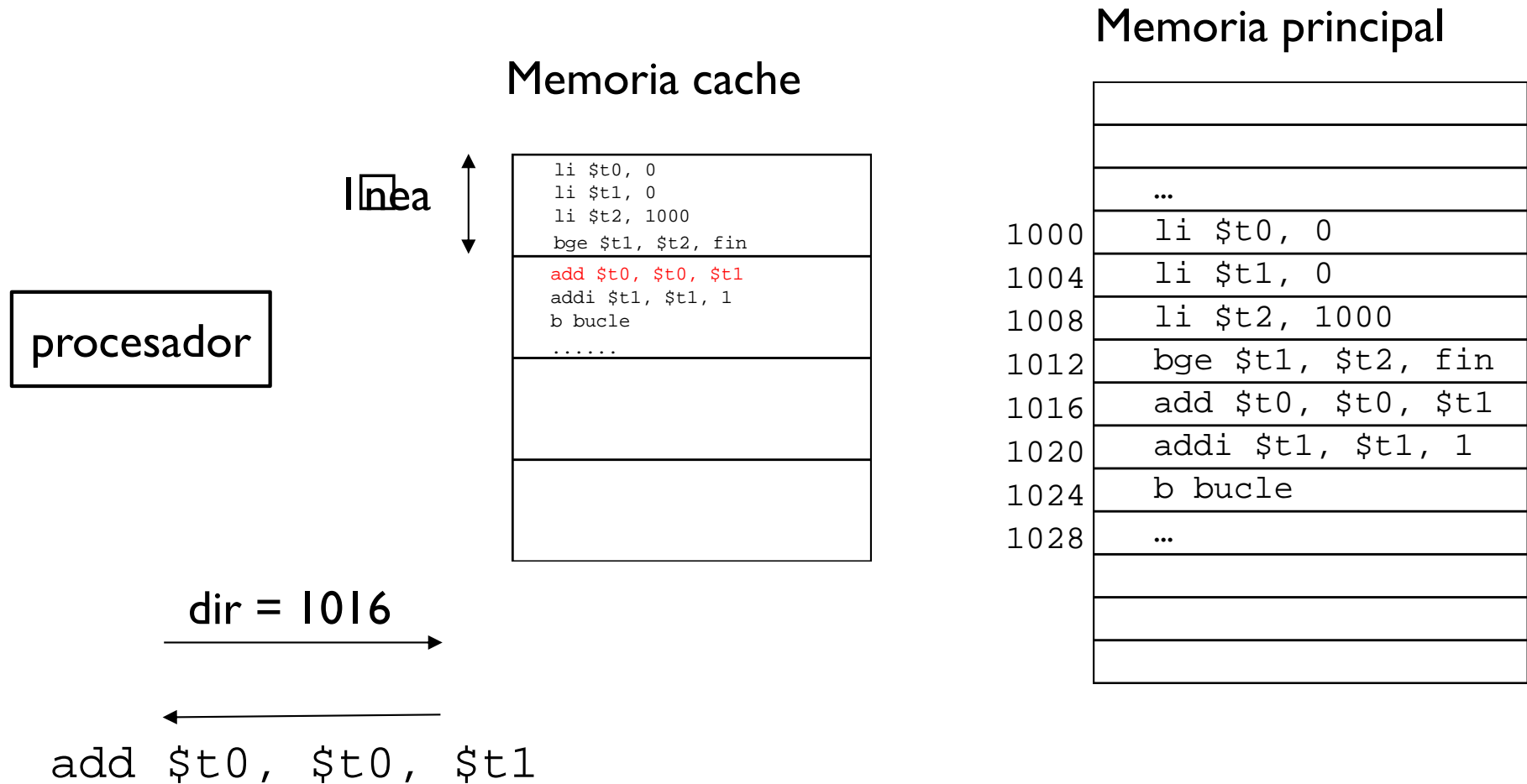
# Ejemplo de funcionamiento



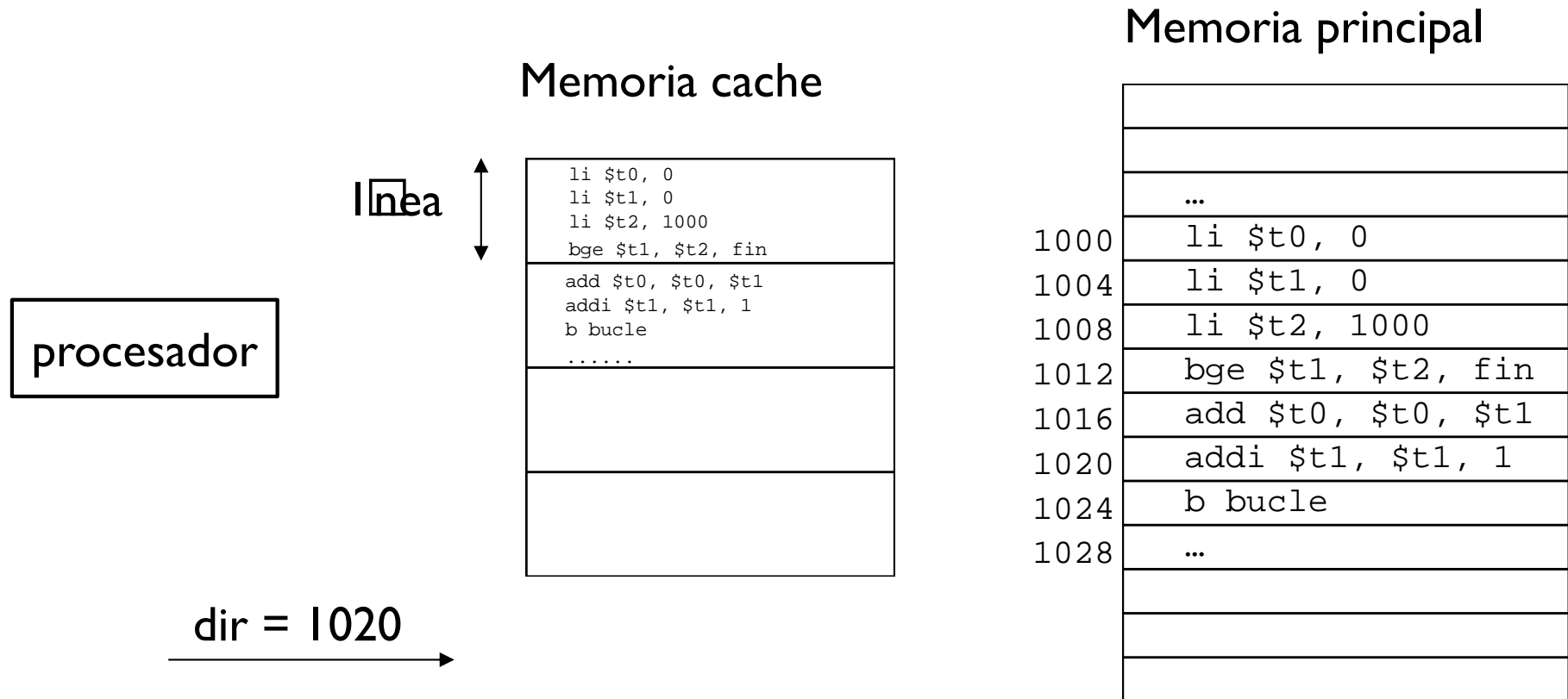
# Ejemplo de funcionamiento



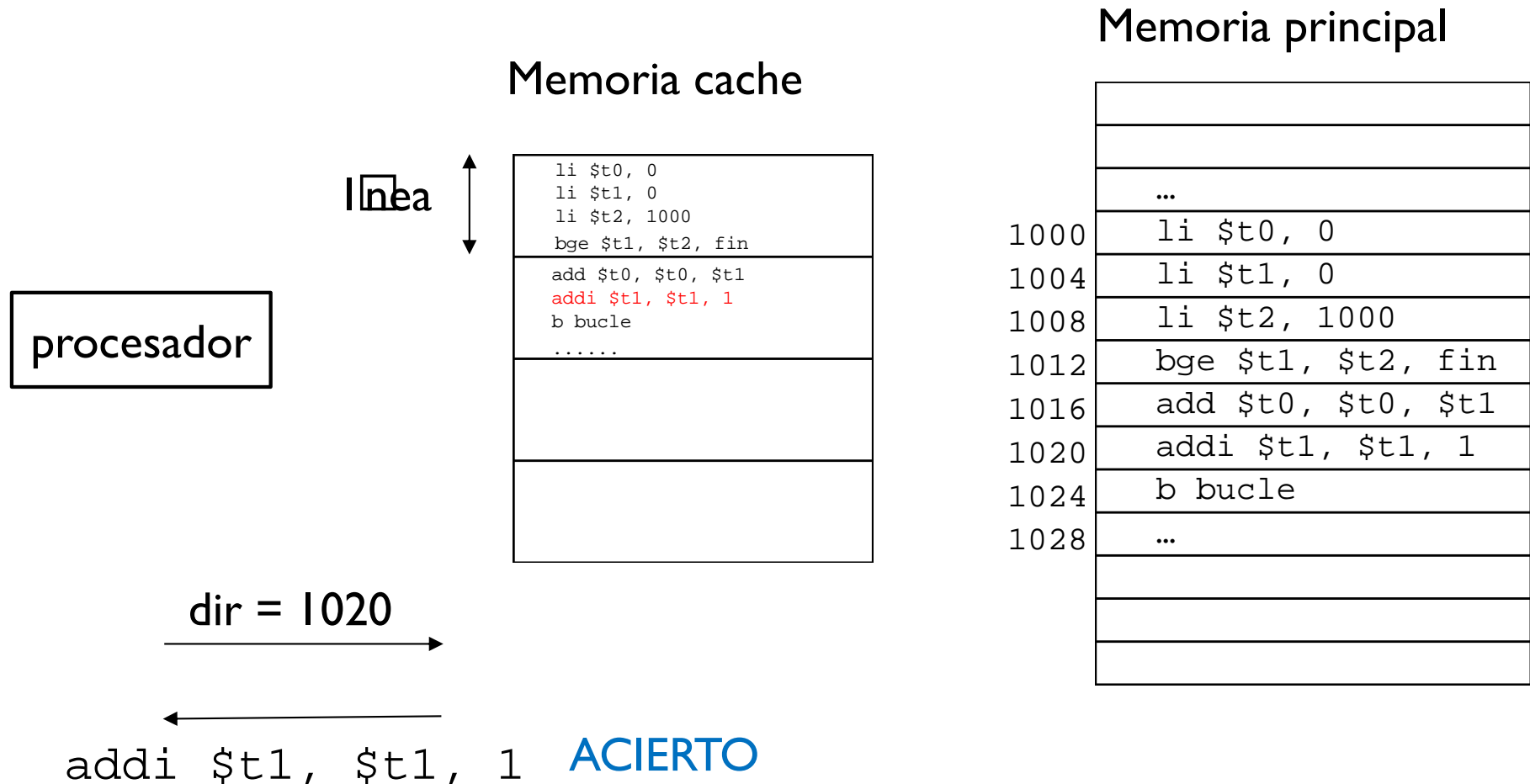
# Ejemplo de funcionamiento



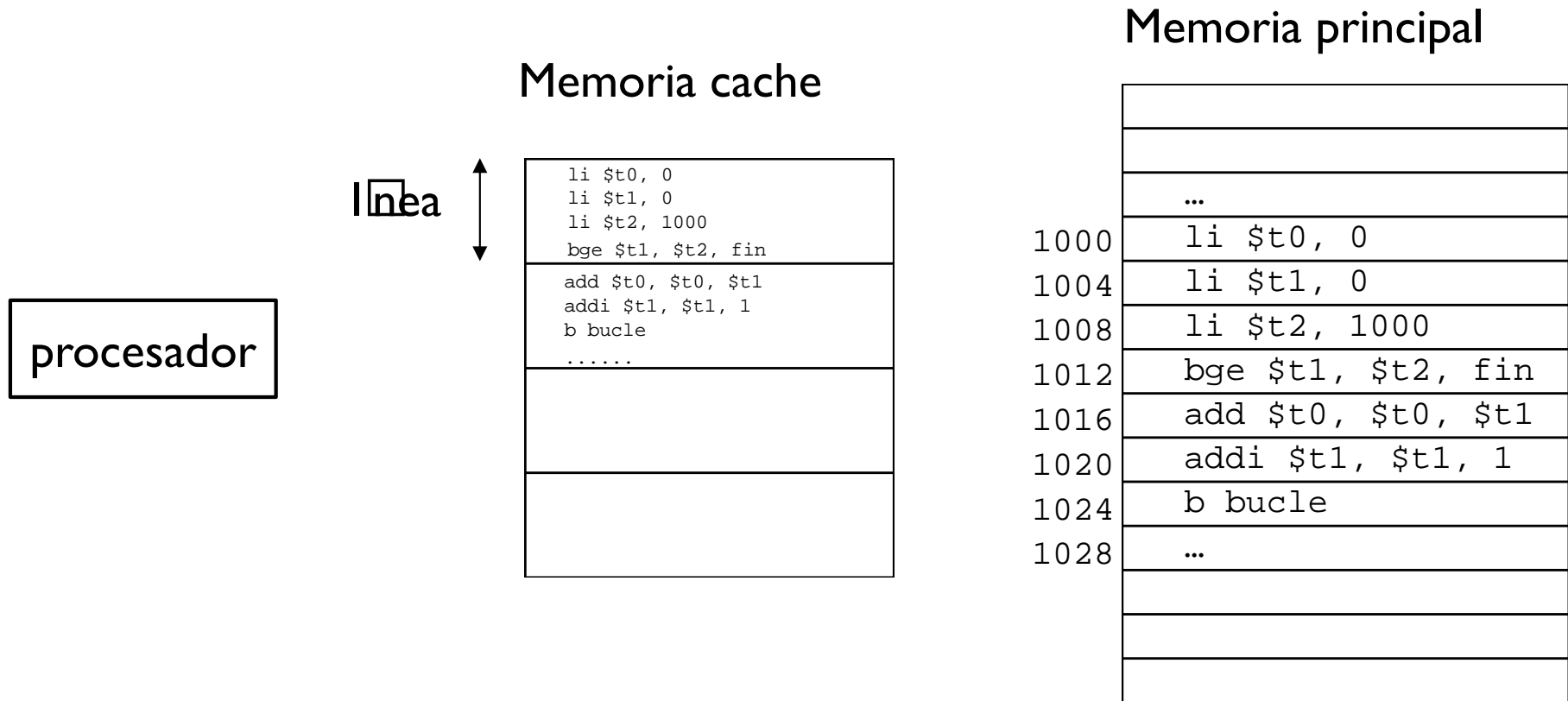
# Ejemplo de funcionamiento



# Ejemplo de funcionamiento



# Ejemplo de funcionamiento



El resto de accesos son **ACIERTOS**

# Ejemplo de funcionamiento

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

bucle:

```
li    $t0, 0    // s  
li    $t1, 0    // i  
li    $t2, 1000  
bge   $t1, $t2, fin
```

fin:

```
add   $t0, $t0, $t1  
addi  $t1, $t1, 1  
b     bucle
```

...

- ▶ Con memoria caché:
  - ▶ Número de bloques = 2



# Ejemplo de funcionamiento

```
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  
b     bucle  
  
fin:  ...
```

► Con memoria caché:

- Número de bloques = 2
- Número de fallos = 2
- Tiempo para transferir 2 bloques =  $200 \times 2 = 400$  ns
- Tiempo de acceso a la caché =  $4004 \times 2 = 8008$  ns
- Tiempo total = 8408 ns

- Acceso a caché: 2 ns
- Acceso a MP: 120 ns
- Bloque de caché: 4 palabras
- Transferencia de un bloque entre memoria principal y caché: 200 ns

# Ejemplo de funcionamiento

```
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  
b     bucle
```

- ▶ Sin memoria caché: 480480 ns
- ▶ Con memoria caché = 8408 ns
- ▶ Tasa de aciertos a la caché =  $4002 / 4004 \Rightarrow 99,95 \%$

# Ejercicio

## Calcular tasa de aciertos

```
int v[1000]; // global      .data:
                               v: .space 4000

int i;
int s;                      .text:
for (i=0; i < 1000; i++)    li    $t0, 0      // i
    s = s + v[i];          li    $t1, 0      // i de v
                               li    $t2, 1000 // componentes
                               li    $t3, 0      // s
bucle: bge    $t0, $t2, fin
        lw    $t4, v($t1)
        addi  $t3, $t3, $t4
        addi  $t0, $t0, 1
        addi  $t1, $t1, 4
        b     bucle
fin:     ...
```

► **Ejemplo:**

- Acceso a caché: 2 ns
- Acceso a MP: 120 ns
- Bloque de MP: 4 palabras
- Transferencia de un bloque entre memoria principal y caché : 200 ns

# ¿Por qué funciona?

- ▶ Tiempo de acceso a caché mucho menor que tiempo de acceso a memoria principal.
- ▶ A la memoria principal se accede por bloques.
- ▶ Cuando un programa accede a una dirección, es probable que vuelva a acceder a ella en el futuro cercano.
  - ▶ Proximidad o Localidad temporal.
- ▶ Cuando un programa accede a una dirección, es probable que en el futuro cercano acceda a posiciones cercanas.
  - ▶ Proximidad o Localidad espacial.
- ▶ Tasa de aciertos: probabilidad de que un dato accedido esté en la caché

**Tasa de aciertos elevada**

# Ejemplos de tiempos de acceso

- ▶ **Memoria principal.**
  - ▶ Tecnología DRAM o similar.
  - ▶ Tiempo de acceso: 20-50 ns.
- ▶ **Memoria caché.**
  - ▶ Tecnología SRAM o similar.
  - ▶ Tiempo de acceso: <1-2.5 ns.

# Tiempo medio de acceso a caché

- ▶ Tiempo medio de acceso a un sistema de memoria con dos niveles

$$\begin{aligned}T_m &= h \cdot T_a + (1-h) \cdot (T_a + T_f) \\ &= T_a + (1-h) \cdot T_f\end{aligned}$$

- ▶  $T_a$ : tiempo de acceso a la caché
- ▶  $T_f$ : tiempo en tratar el fallo, tiempo en remplazar un bloque y traerlo de memoria principal a caché
- ▶  $h$ : tasa de aciertos

# Rendimiento general

$$\begin{aligned}T_m &= h \cdot T_a + (1-h) \cdot (T_a + T_f) \\ &= T_a + (1-h) \cdot T_f\end{aligned}$$

1. El procesador realiza un acceso a memoria caché.
2. La cache comprueba si ya están los datos de esta posición:
  - ▶ **Si está (ACIERTO),**
    - 3.A.1 Se la sirve al procesador desde la cache (rápidamente):  $T_a$
  - ▶ **Si no está (FALLO),**
    - 3.B.1 La cache transfiere de memoria principal el bloque asociado a la posición:  $T_f$
    - 3.B.2 Después, la cache entrega los datos pedidos al procesador :  $T_a$

# Ejemplo

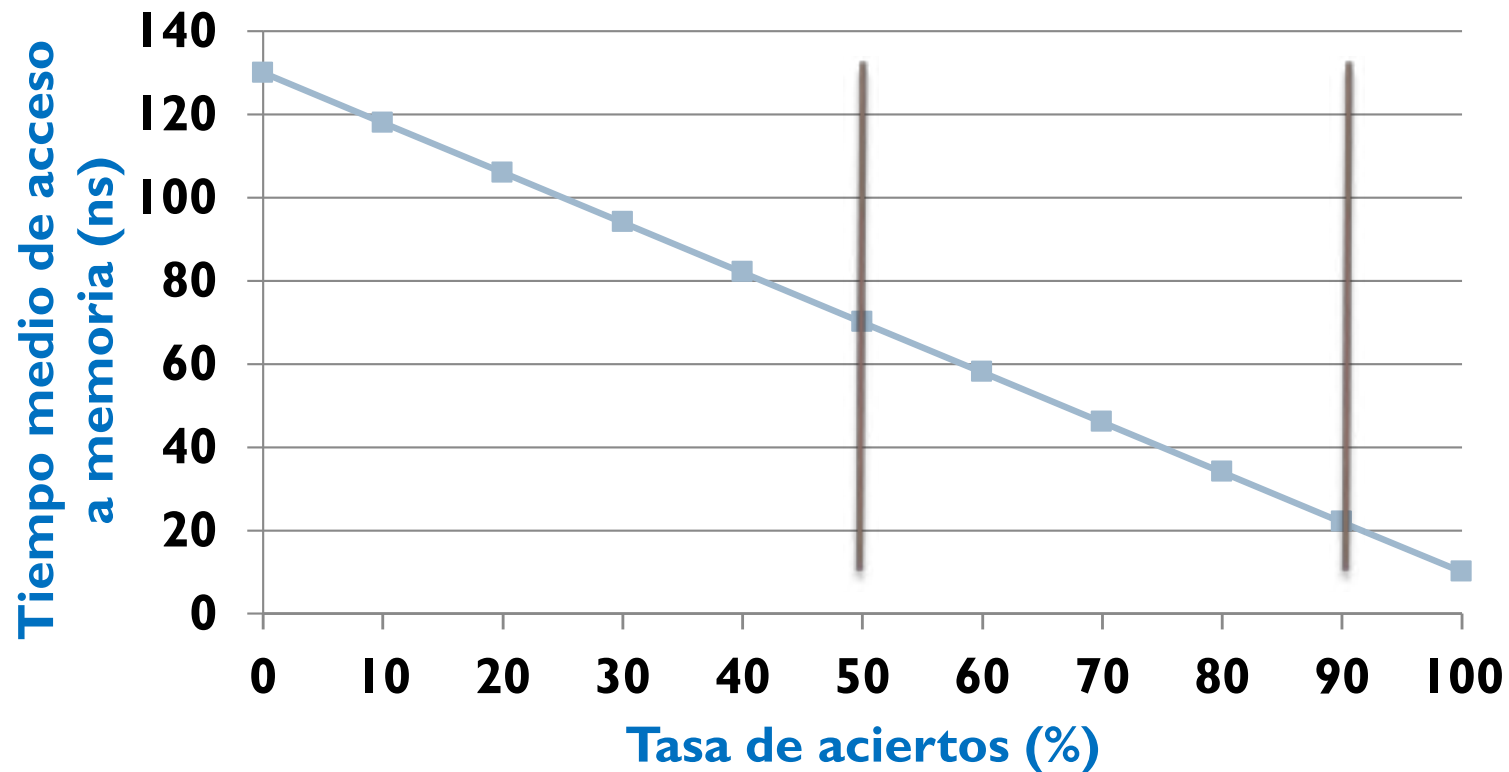
$$\begin{aligned}T_m &= h \cdot T_a + (1-h) \cdot (T_a + T_f) \\ &= T_a + (1-h) \cdot T_f\end{aligned}$$

1.  $T_a$ : Tiempo de acceso a caché = **10 ns**
2.  $T_f$ : Tiempo de acceso a memoria principal = **120 ns**
3.  $h$ : tasa de aciertos  $\rightarrow X = 0.1, 0.2, \dots, 0.9, 1.0$   
**10%, 20%, ..., 90%, 100%**



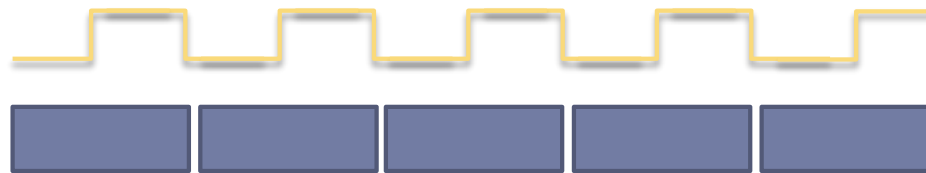
# Ejemplo

$$\begin{aligned} T_m &= h \cdot T_a + (1-h) \cdot (T_a + T_f) \\ &= T_a + (1-h) \cdot T_f \end{aligned}$$



# Acceso por bloques

- ▶ Se premia el acceso a posiciones consecutivas de memoria
  - ▶ Ejemplo 1:  
acceder a 5 posiciones de memoria **individuales (no consecutivas)**

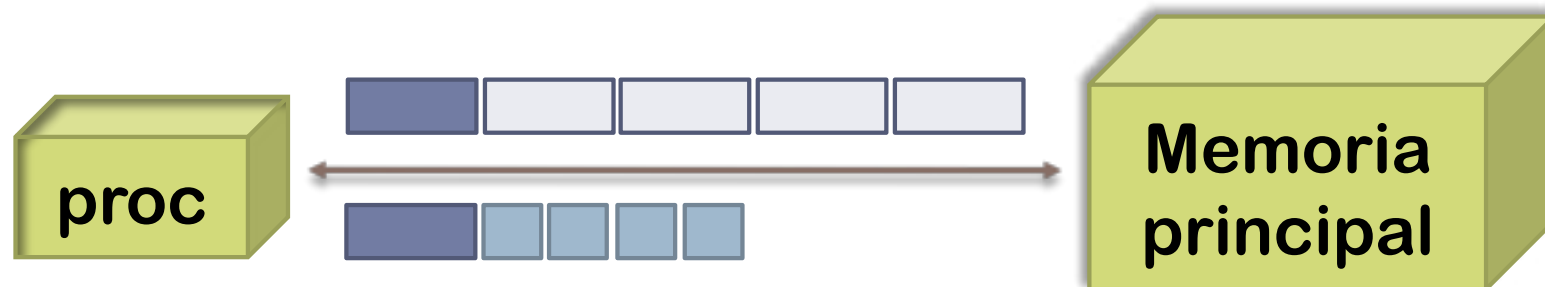


- ▶ Ejemplo 2:  
acceder a 5 posiciones de memoria **consecutivas**



# Acceso por bloques

- ▶ Si se accede a un conjunto consecutivo de posiciones de memoria, los accesos siguientes al primero tienen un coste menor.



# Niveles de memoria caché

- ▶ Es habitual encontrar tres niveles:
  - ▶ **L1 o nivel 1:**
    - ▶ Caché interna: la más cercana al procesador
    - ▶ Tamaño pequeño (8KB-128KB) y máxima velocidad
    - ▶ Pueden separarse para instrucciones y datos
  - ▶ **L2 o nivel 2:**
    - ▶ Caché interna
    - ▶ Entre L1 y L3 (o entre L1 y M.Principal)
    - ▶ Tamaño mediano (256KB – 4MB) y menor velocidad que L1
  - ▶ **L3 o nivel 3:**
    - ▶ Tipicamente último nivel antes de M. Principal
    - ▶ Tamaño mayor y menor velocidad que L2
    - ▶ Interna o externa al procesador

# Ejercicio

- ▶ **Computador:**
  - ▶ Tiempo de acceso a caché: 4 ns
  - ▶ Tiempo de acceso a bloque de MP: 120 ns.
- ▶ Si se tiene una tasa de aciertos del 90%. ¿Cuál es el tiempo medio de acceso?
- ▶ Tasas de acierto necesarias para que el tiempo medio de acceso sea menor de 10 ns y 5 ns.

# Ejercicio (solución)

- ▶ **Computador:**
  - ▶ Tiempo de acceso a caché: 4 ns
  - ▶ Tiempo de acceso a bloque de MP: 120 ns.
- ▶ Si se tiene una tasa de aciertos del 90%. ¿Cuál es el tiempo medio de acceso?

$$T_m = 4 \times 0.9 + (120 + 4) \times 0.1 = 16 \text{ ns}$$

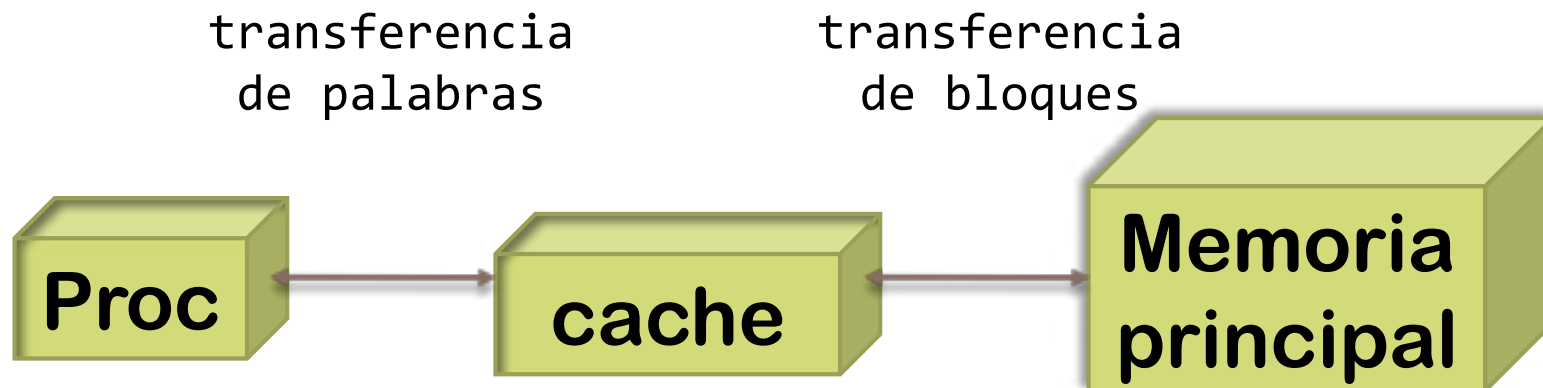
- ▶ Tasas de acierto necesarias para que el tiempo medio de acceso sea menor de 5 ns.

$$\begin{aligned} 5 &= 4 \times h + (120 + 4) \times (1 - h) = 16 \text{ ns} \Rightarrow \\ \Rightarrow h &> 0.9916 \end{aligned}$$

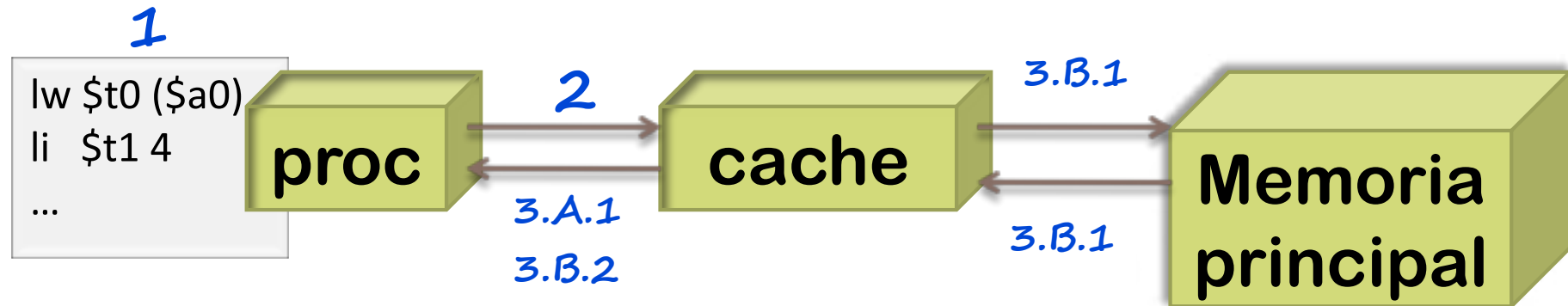
# Memoria caché

## Resumen

- ▶ Se construye con tecnología SRAM
  - ▶ Integrada en el mismo procesador.
  - ▶ Más rápida y más cara que la memoria DRAM.
- ▶ Mantiene una **copia** de partes de la memoria principal.



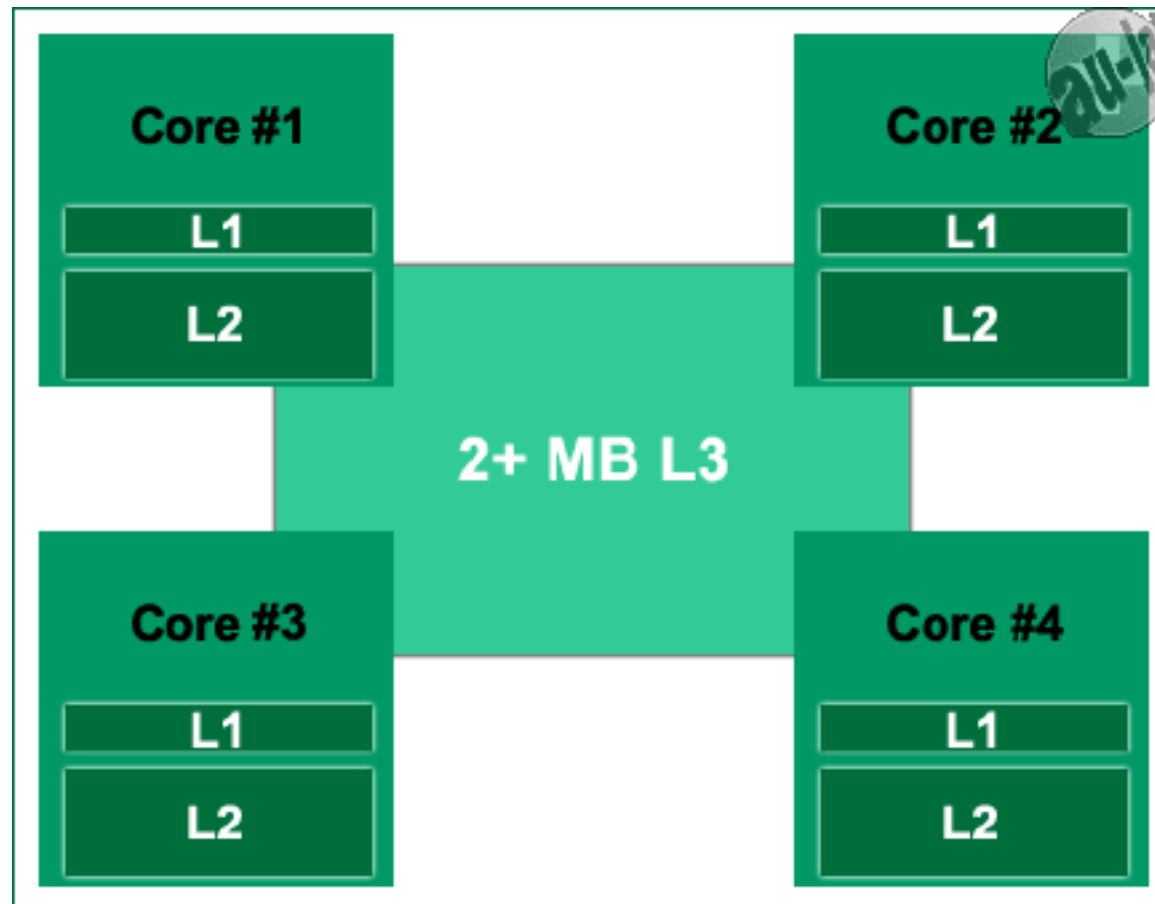
# Funcionamiento general



1. El procesador solicita contenidos de una posición (dirección) de memoria.
2. La cache comprueba si ya están los datos de esta posición:
  - Si está (**AICERTO**),
    - 3.A.1 Se la sirve al procesador desde la cache (rápidamente).
  - Si no está (**FALLO**),
    - 3.B.1 La cache transfiere de memoria principal el bloque de palabras asociado a la posición.
    - 3.B.2 Después, la cache entrega los datos pedidos a la procesador.

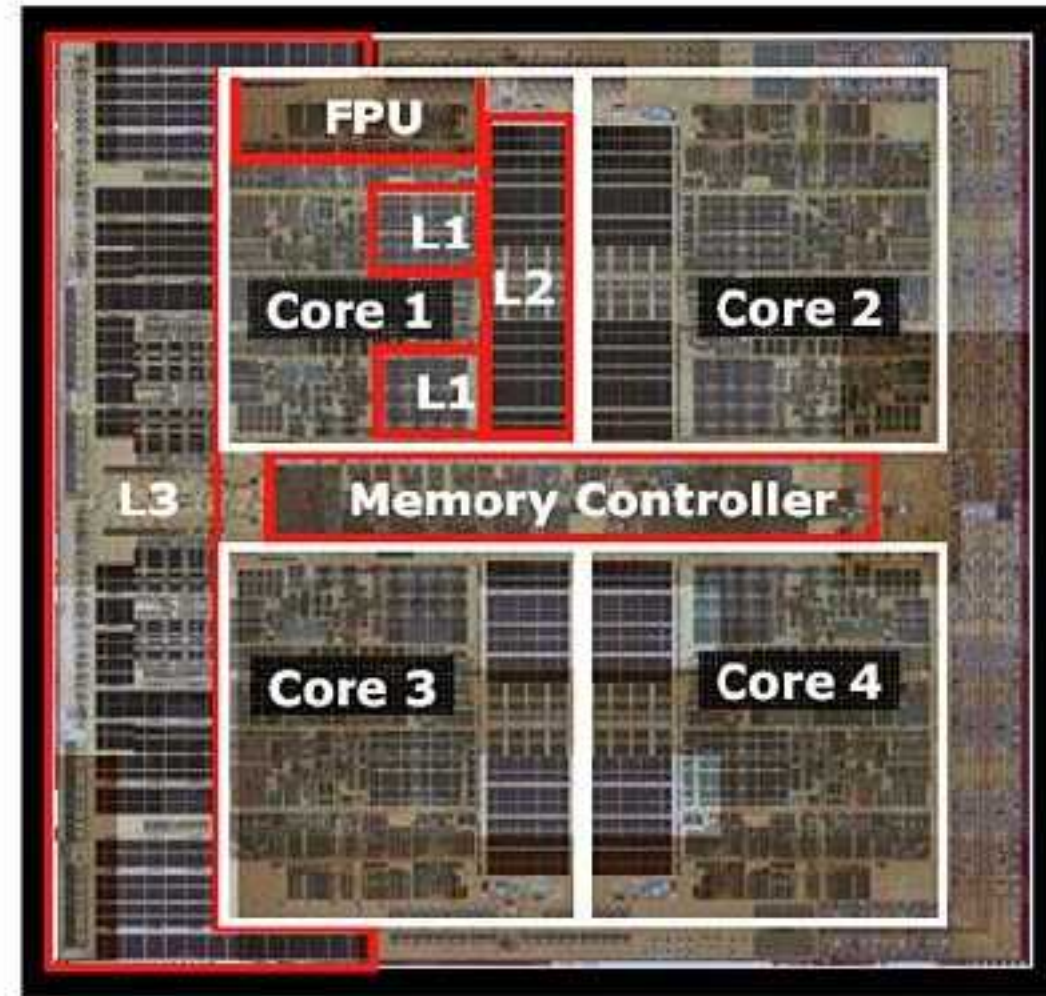


# Ejemplo: AMD quad-core

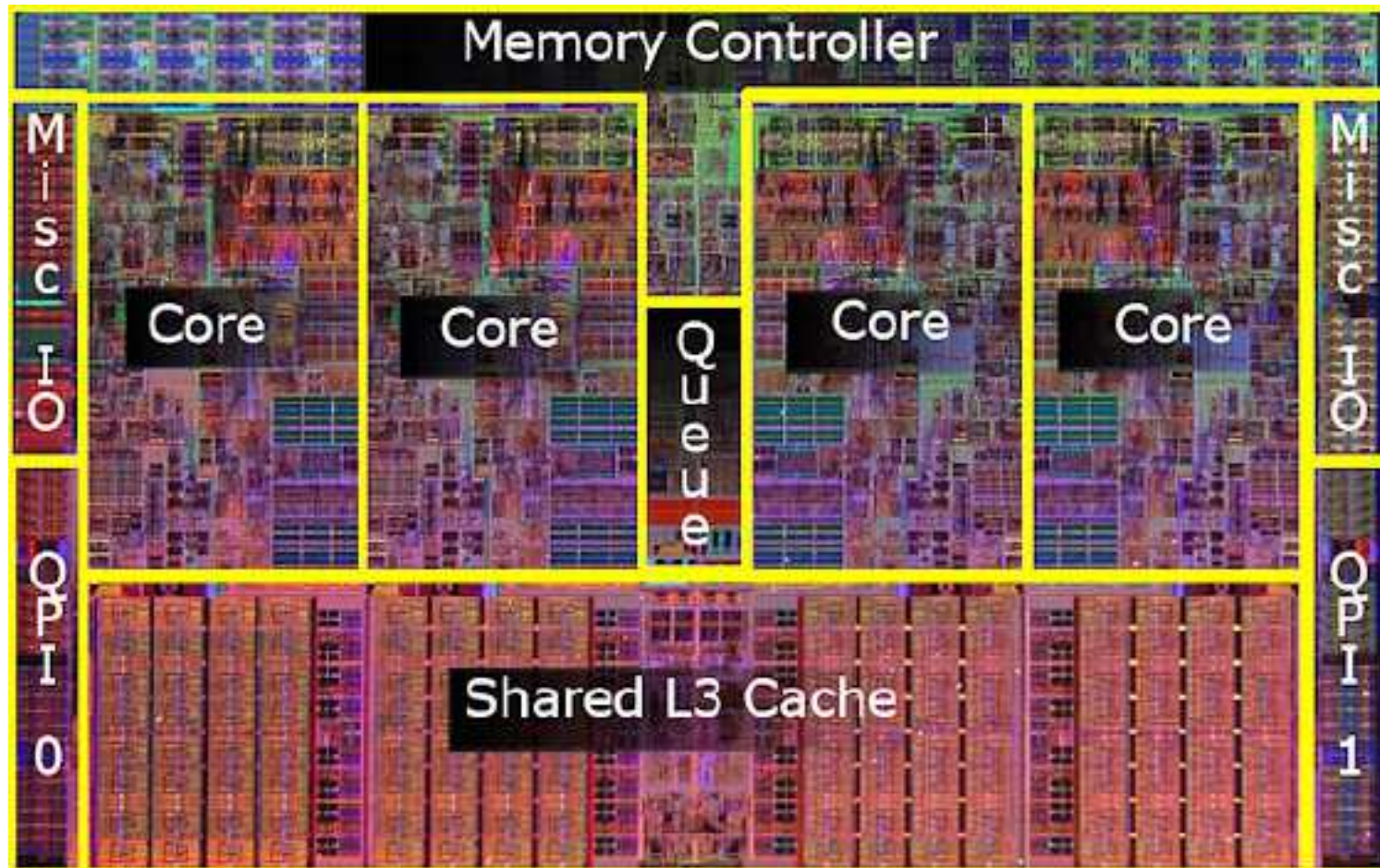


Quad-Core CPU mit gemeinsamen L3-Cache

# Ejemplo: AMD quad-core



# Ejemplo: Intel Core i7



# Diseño y organización de la memoria caché

## 1. Estructura de la memoria caché

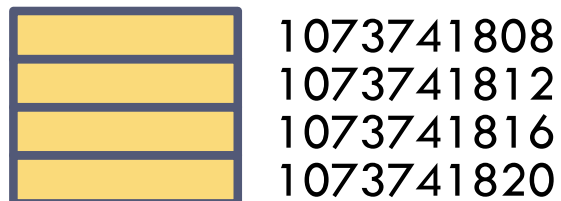
## 2. Diseño de la memoria cache

- ▶ Tamaño
- ▶ Función de correspondencia
- ▶ Algoritmo de sustitución
- ▶ Política de escritura

# Organización de la memoria principal



...

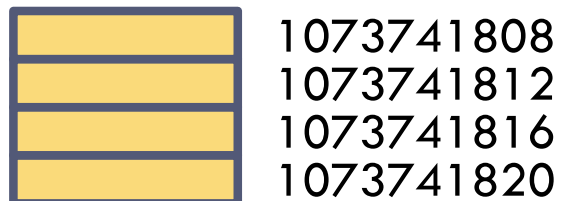


- ▶ La memoria principal *lógicamente* se divide en bloques de igual tamaño.
  - ▶ 1 bloque = k palabras.
- ▶ Ejercicio: ¿Cuántos bloques de 4 palabras hay en una memoria de 1 GB?

# Organización de la memoria principal



...



- ▶ La memoria principal *lógicamente* se divide en bloques de igual tamaño.

- ▶ 1 bloque = k palabras.

- ▶ Ejercicio: ¿Cuántos bloques de 4 palabras hay en una memoria de 1 GB?

- ▶ *Solución:*

$$2^{30} \text{ B} / 16 \text{ B} = 2^{30-4} \text{ B} = 2^{26} \text{ B} = 64 \text{ megabloques} \approx 64 \text{ millones}$$



# Acceso a una palabra en memoria principal

- ▶ Ejemplo:
  - ▶ Computador de 32 bits
  - ▶ Memoria direccionada por bytes
  - ▶ Acceso a memoria principal por palabras
  - ▶ Acceso a la palabra con dirección

**0x00000064**

← Palabra de 32 bits →

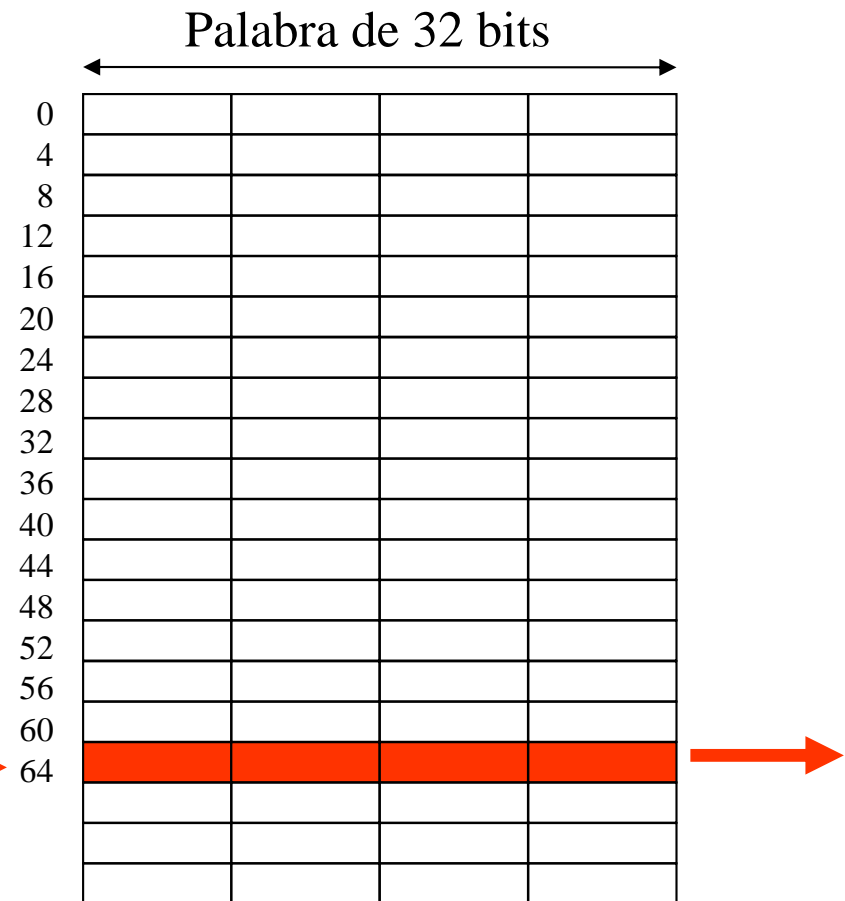
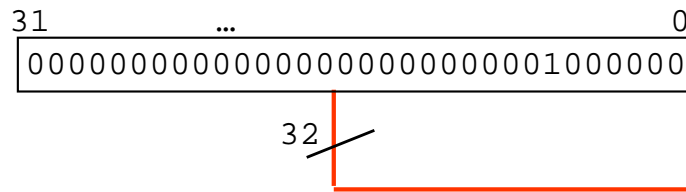
0				
4				
8				
12				
16				
20				
24				
28				
32				
36				
40				
44				
48				
52				
56				
60				
64				

# Acceso a una palabra en memoria principal

► Ejemplo:

- ▶ Computador de 32 bits
- ▶ Memoria direccionada por bytes
- ▶ Acceso a memoria principal por palabras
- ▶ Acceso a la palabra con dirección

0x00000064





# Organización de la memoria caché

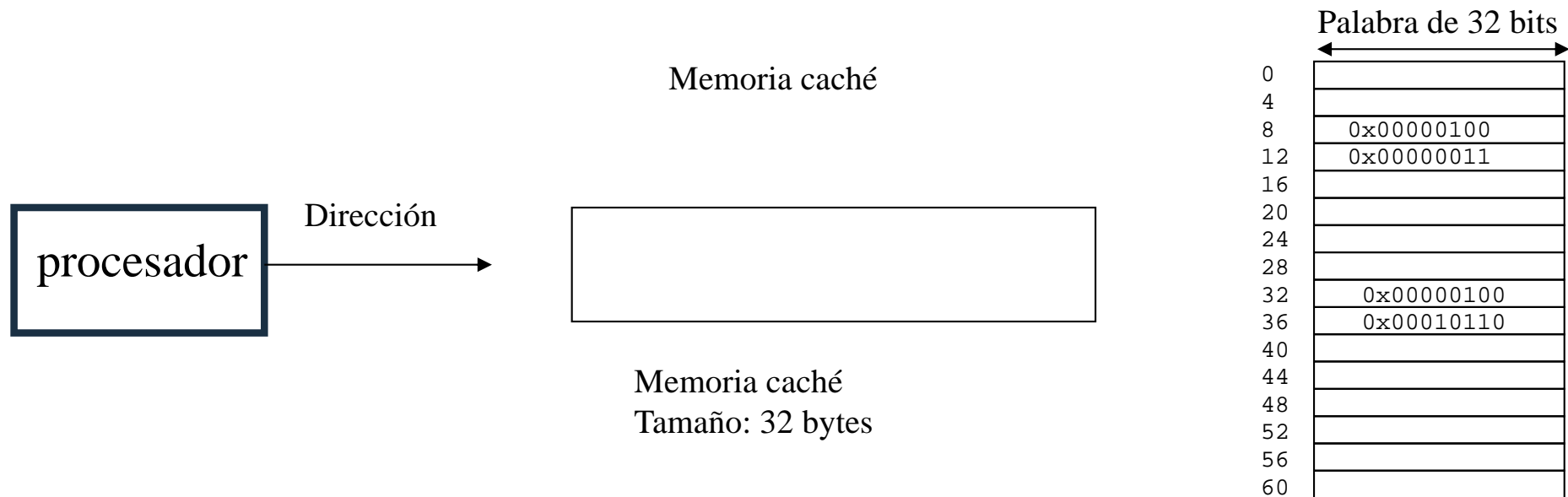
- ▶ La memoria caché está organizada en bloques (**líneas**)
  - ▶ Al bloque de memoria caché se le llama **línea** de caché
- ▶ El tamaño del bloque de M. principal es igual al tamaño de la **línea**.
  - ▶ Pero el tamaño de la memoria caché es mucho menor.
  - ▶ El número de bloques que cabe en la memoria caché es muy pequeño.
- ▶ ¿Cuántos bloques de 4 palabras cabe en una memoria caché de 32 KB?

# Organización de la memoria caché

- ▶ La memoria caché está organizada en bloques (líneas)
  - ▶ Al bloque de memoria caché se le llama línea de caché
- ▶ El tamaño del bloque de M. principal es igual al tamaño de la línea.
  - ▶ Pero el tamaño de la memoria caché es mucho menor.
  - ▶ El número de bloques que cabe en la memoria caché es muy pequeño.
- ▶ ¿Cuántos bloques de 4 palabras cabe en una memoria caché de 32 KB?
  - ▶ Solución:
$$2^5 \cdot 2^{10} \text{ B} / 2^4 \text{ B} = 2^{11} \text{ bloques} = 2048 \text{ bloques} = 2048 \text{ líneas}$$

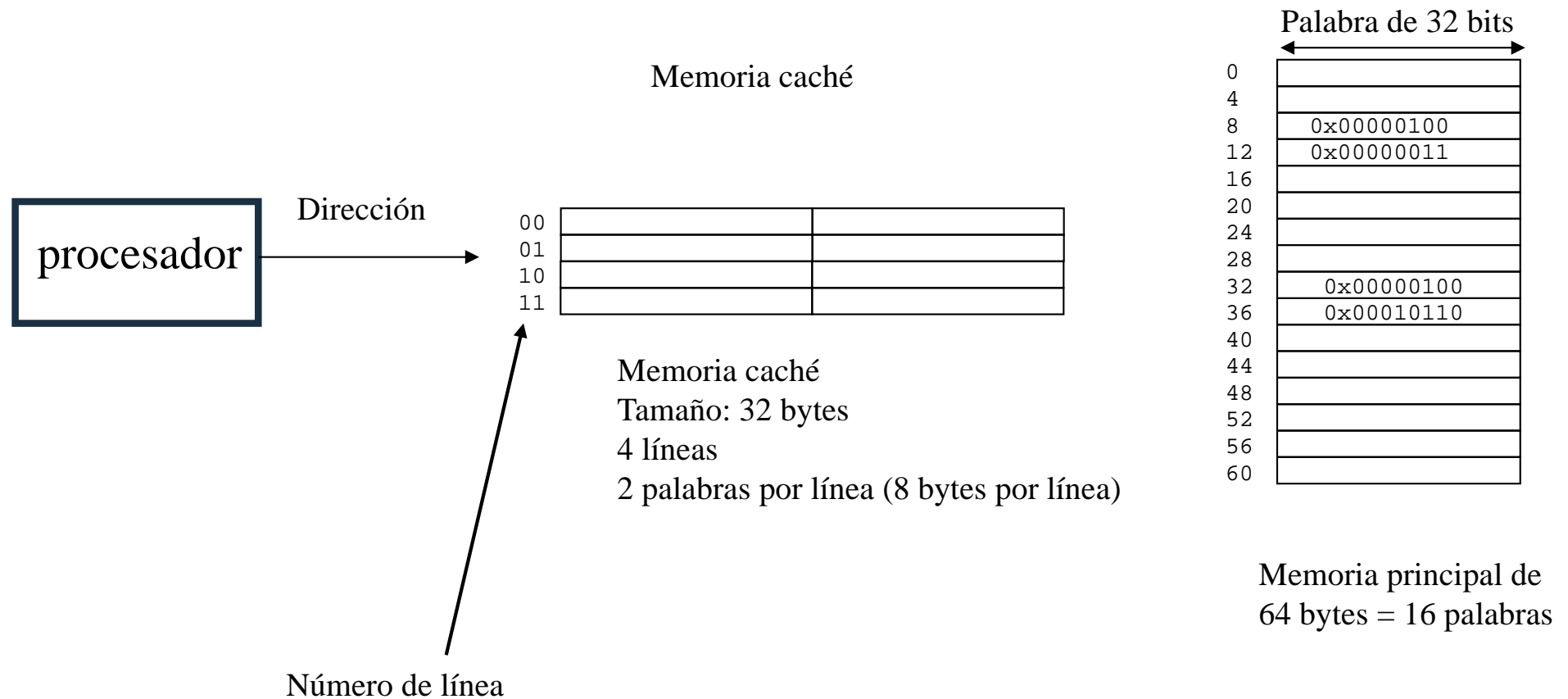
# ¿Cómo buscar una palabra en caché?

**Ejemplo:**

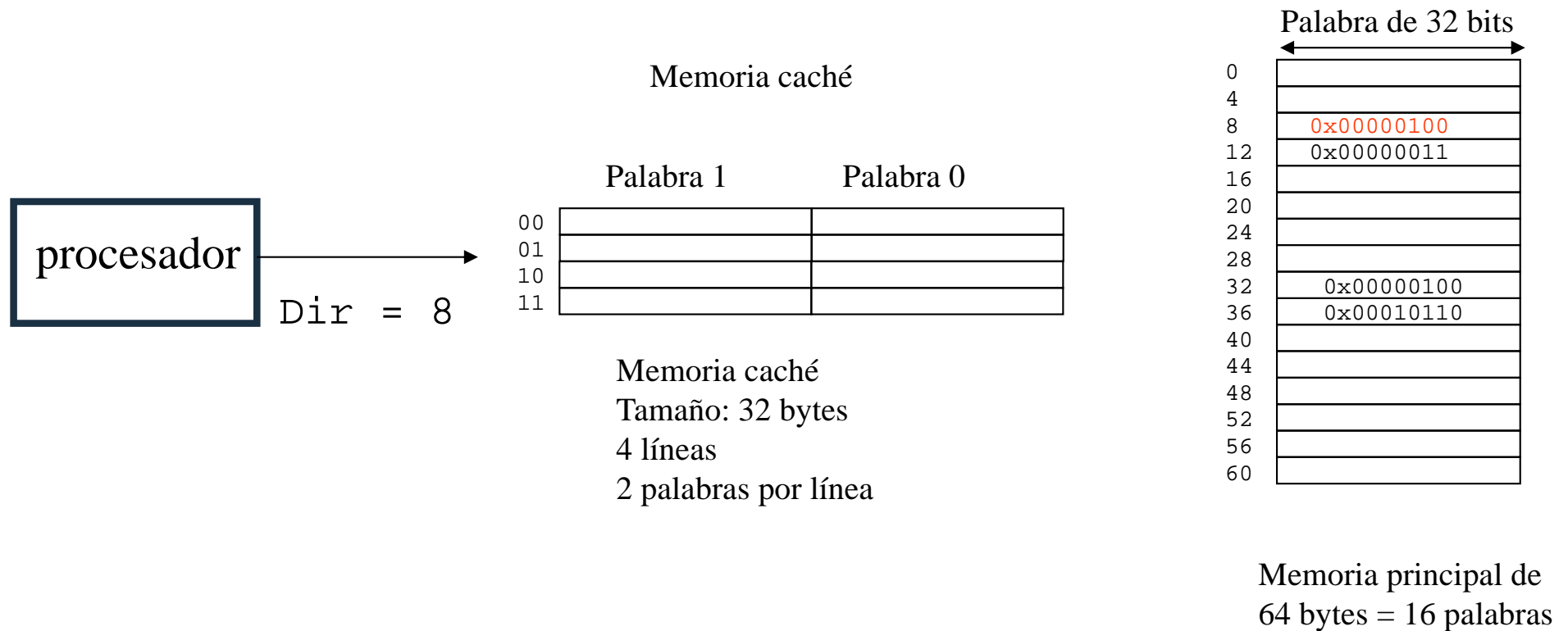


Con líneas de dos palabras  
¿cuántas líneas tiene la caché?

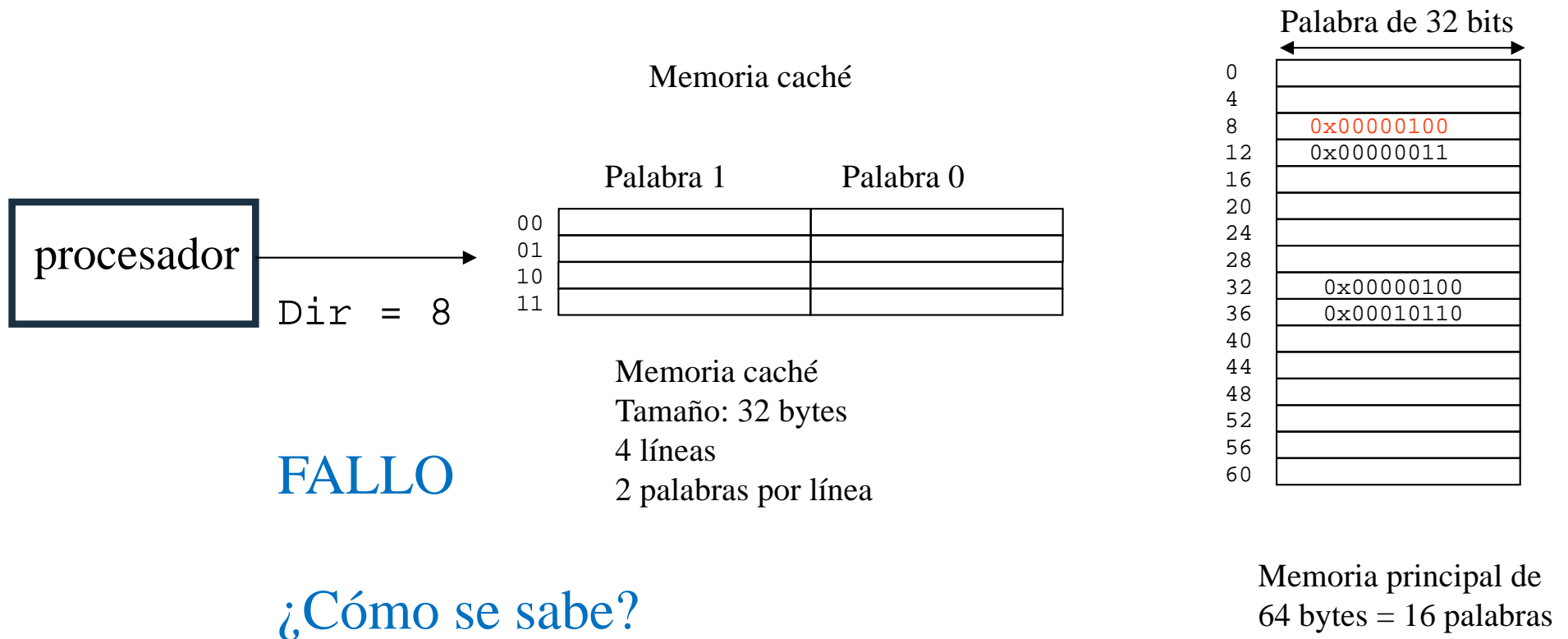
# ¿Cómo buscar una palabra en la cache?



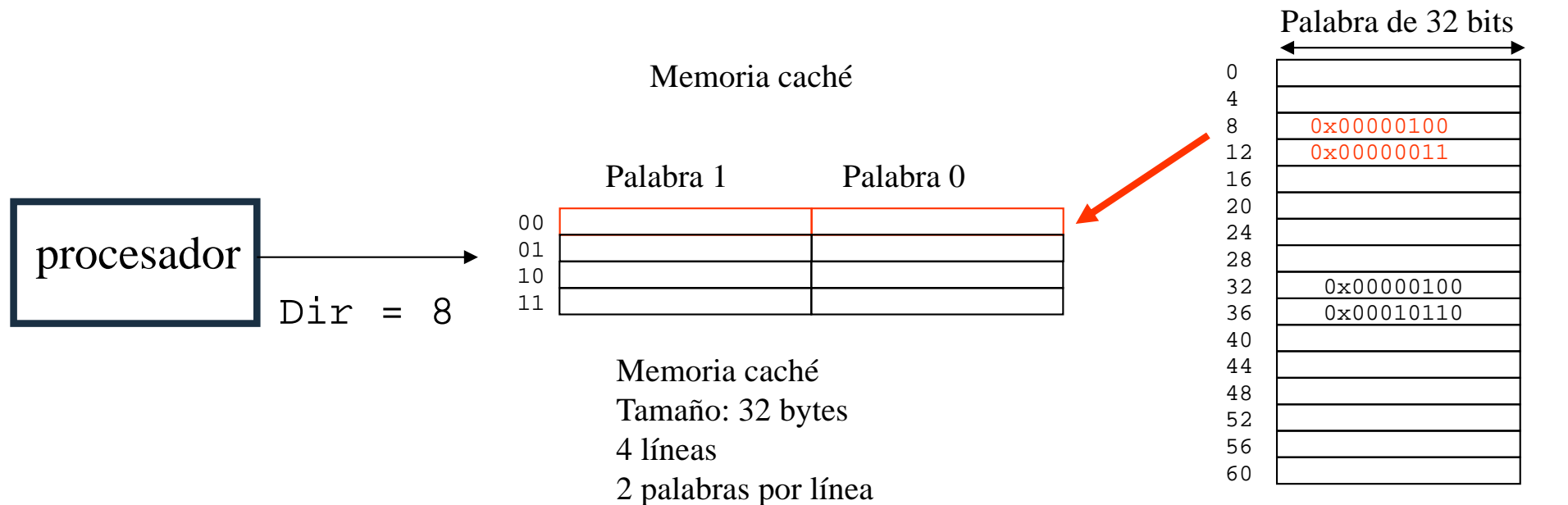
# ¿Cómo buscar una palabra en caché?



# ¿Cómo buscar una palabra en caché?



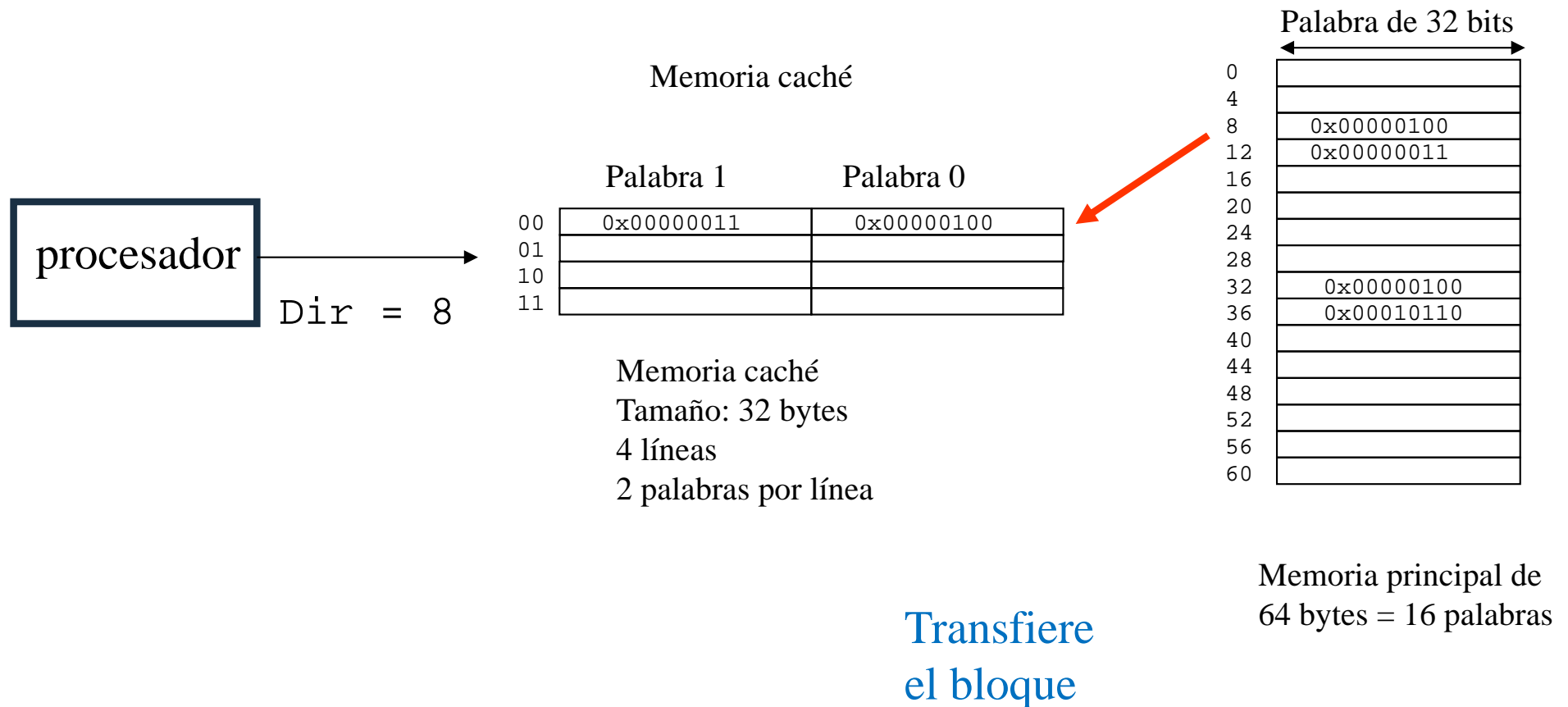
# ¿Cómo buscar una palabra en caché?



Se elige una  
línea en la caché  
¿Qué línea?

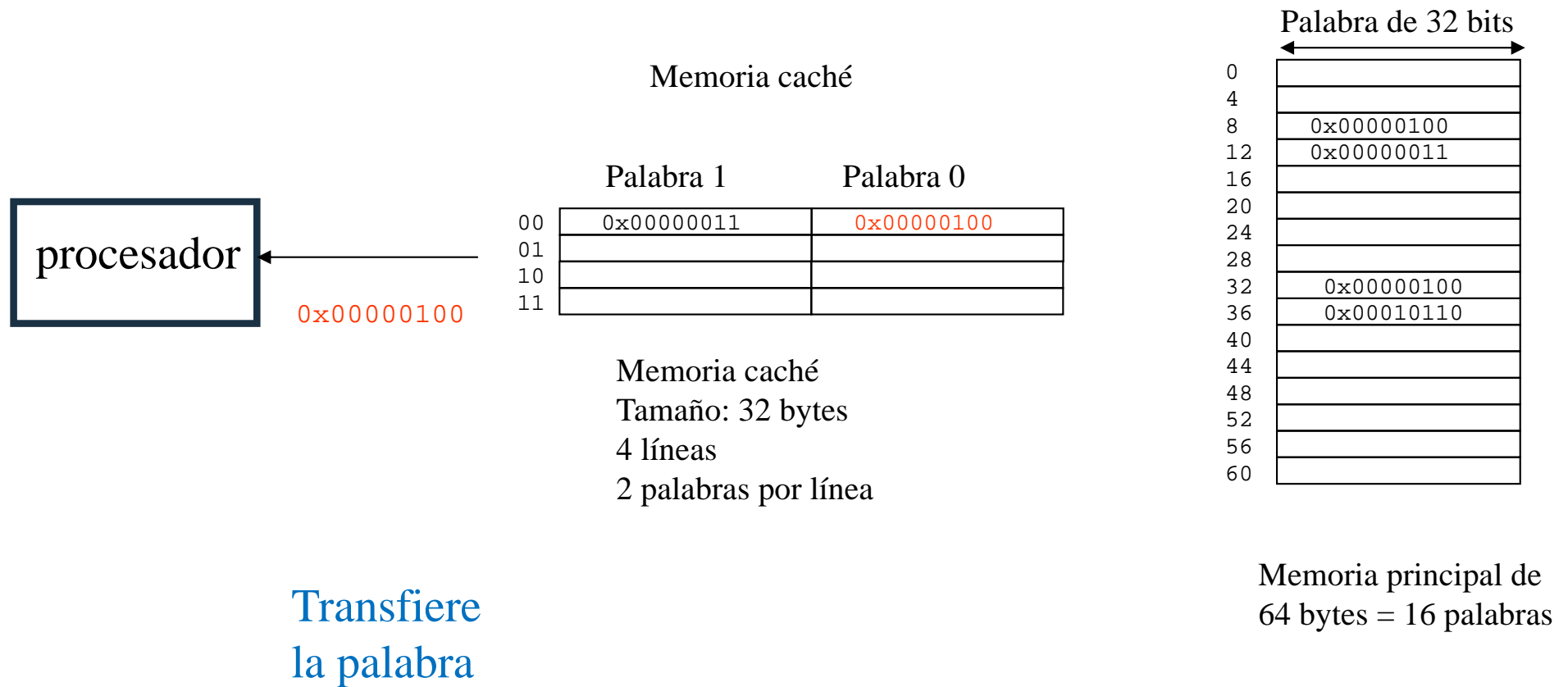
Memoria principal de  
64 bytes = 16 palabras

# ¿Cómo buscar una palabra en caché?

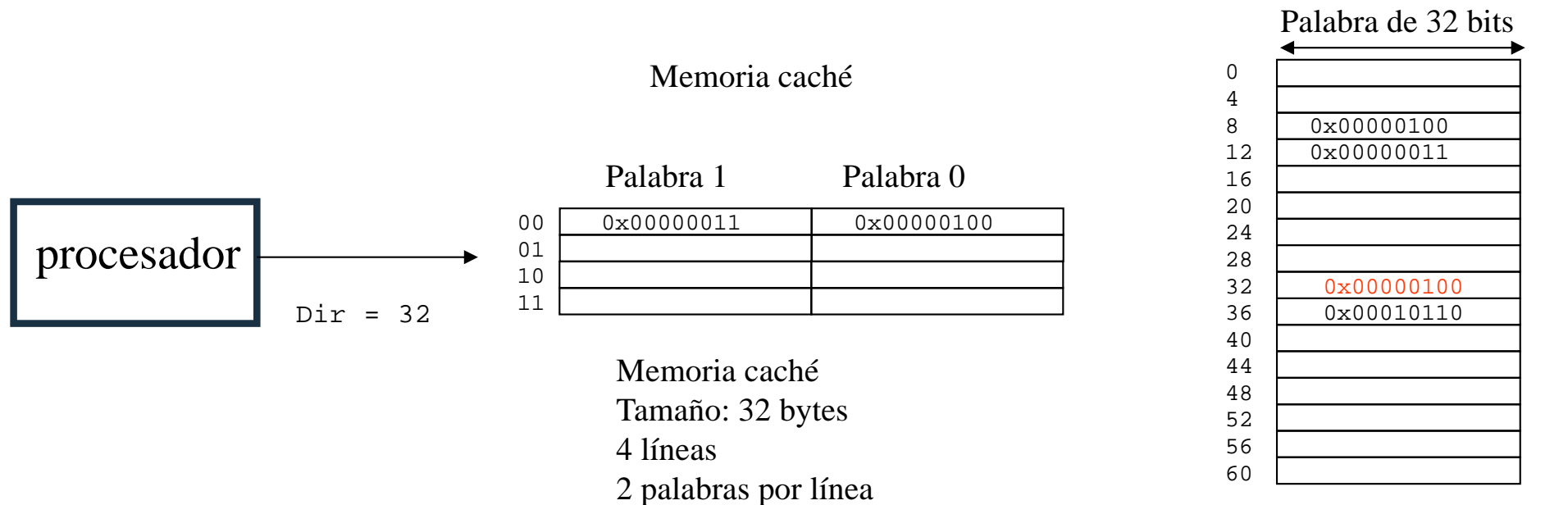




# ¿Cómo buscar una palabra en caché?



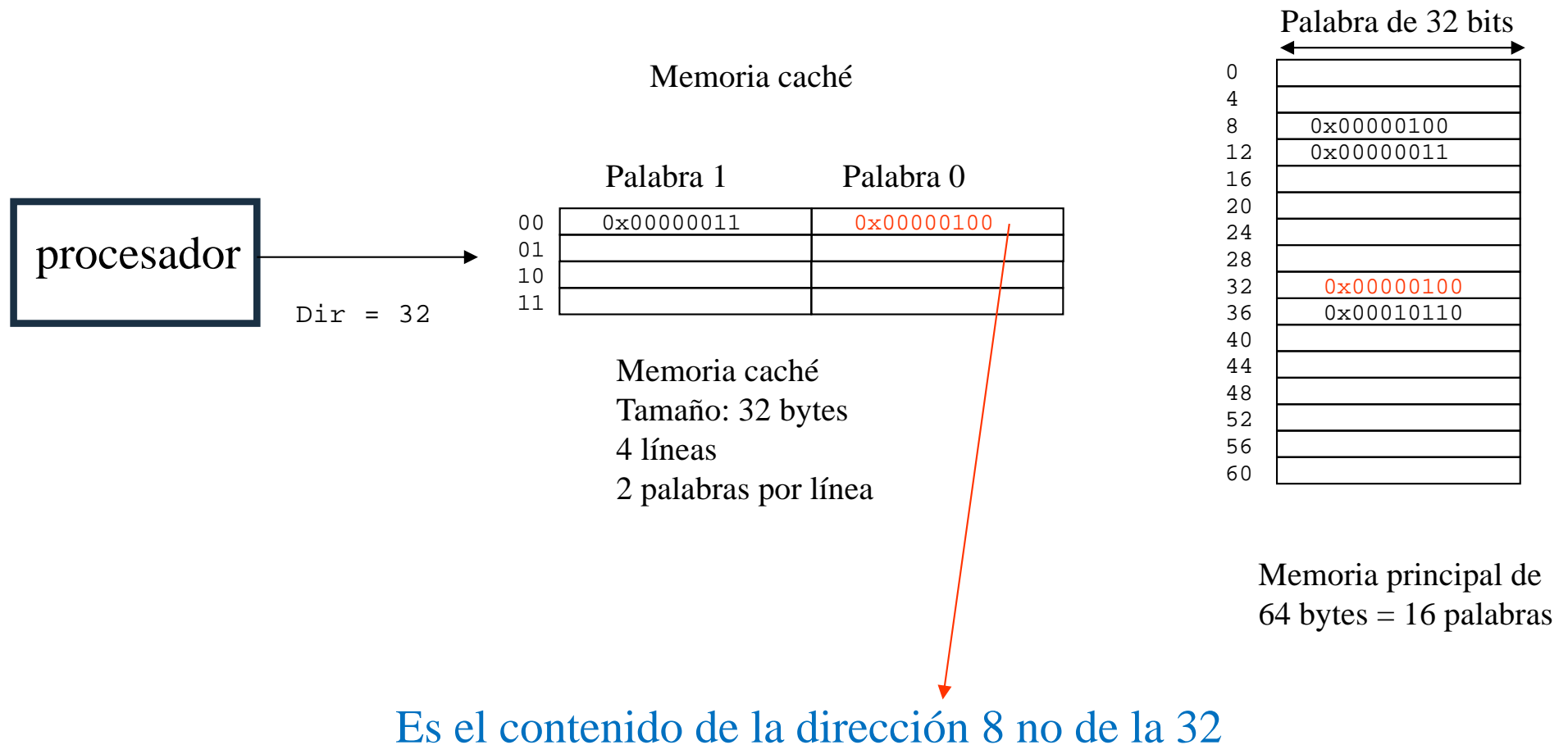
# ¿Cómo buscar una palabra en caché?



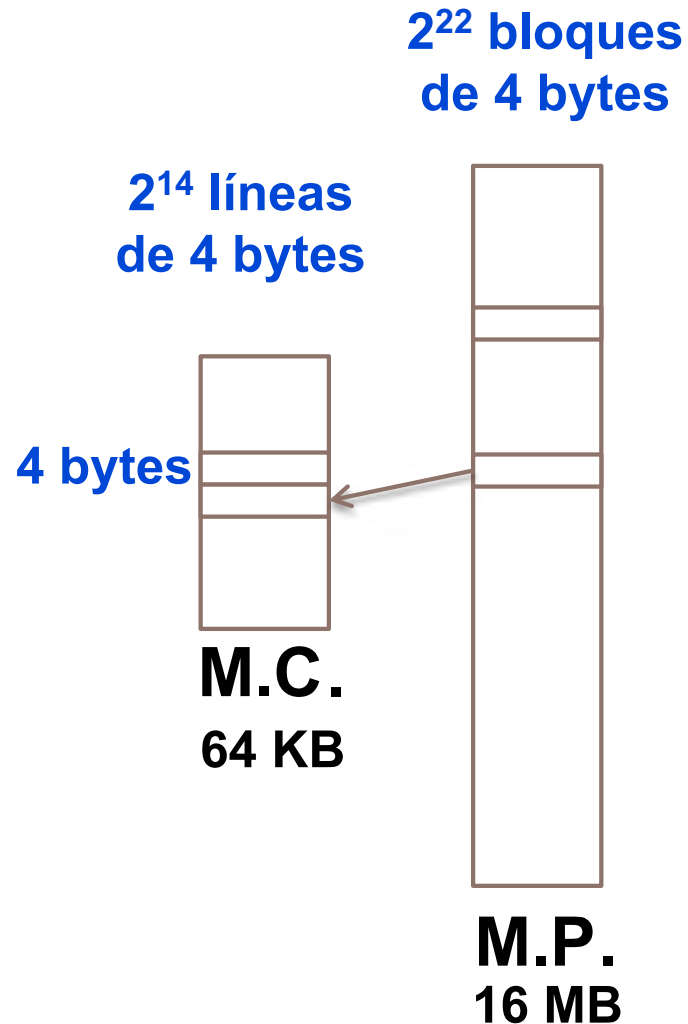
Memoria principal de  
64 bytes = 16 palabras

¿Cómo saber si está en la caché?

# ¿Cómo buscar una palabra en caché?

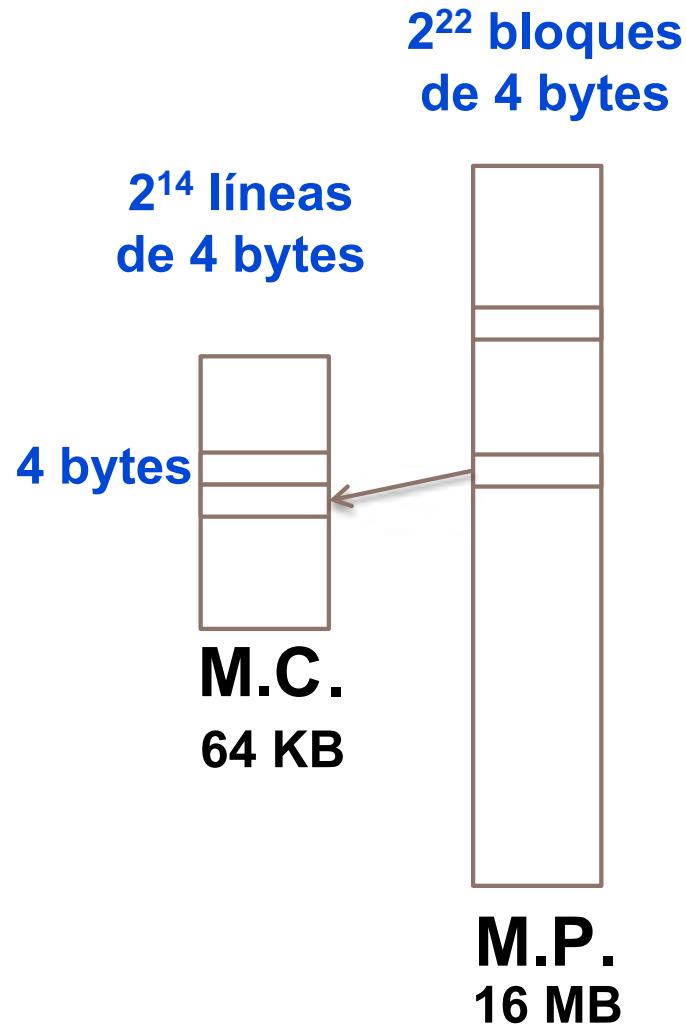


# Estructura y diseño de la cache



- ▶ Se divide la M.P. y la M.C. en bloques de igual tamaño
- ▶ A cada bloque de M.P. le corresponderá **una línea de M.C.** (bloque en caché)

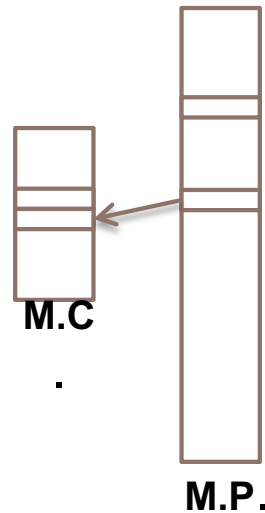
# Estructura y diseño de la cache



- ▶ Se divide la M.P. y la M.C. en bloques de igual tamaño
- ▶ A cada bloque de M.P. le corresponderá **una línea de M.C.** (bloque en caché)
- ▶ En el diseño se determina:
  - ▶ **Tamaño**
  - ▶ **Función de correspondencia**
  - ▶ **Algoritmo de sustitución**
  - ▶ **Política de escritura**
- ▶ Es habitual distintos diseños para L1, L2, ...

# Tamaño de caché

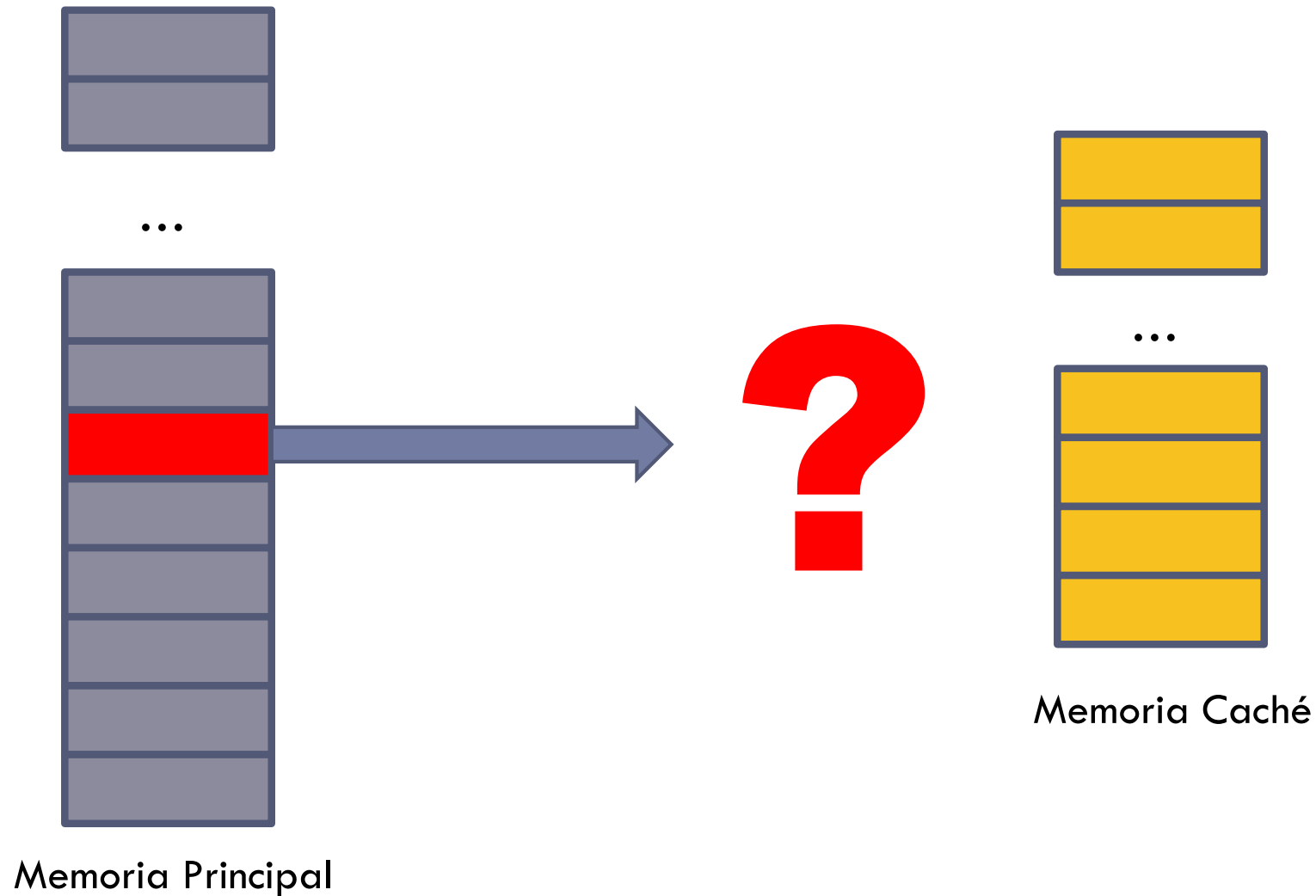
- ▶ El tamaño total y de las líneas en los que se organiza
- ▶ Se determina por estudios sobre códigos muy usados



# Función de correspondencia

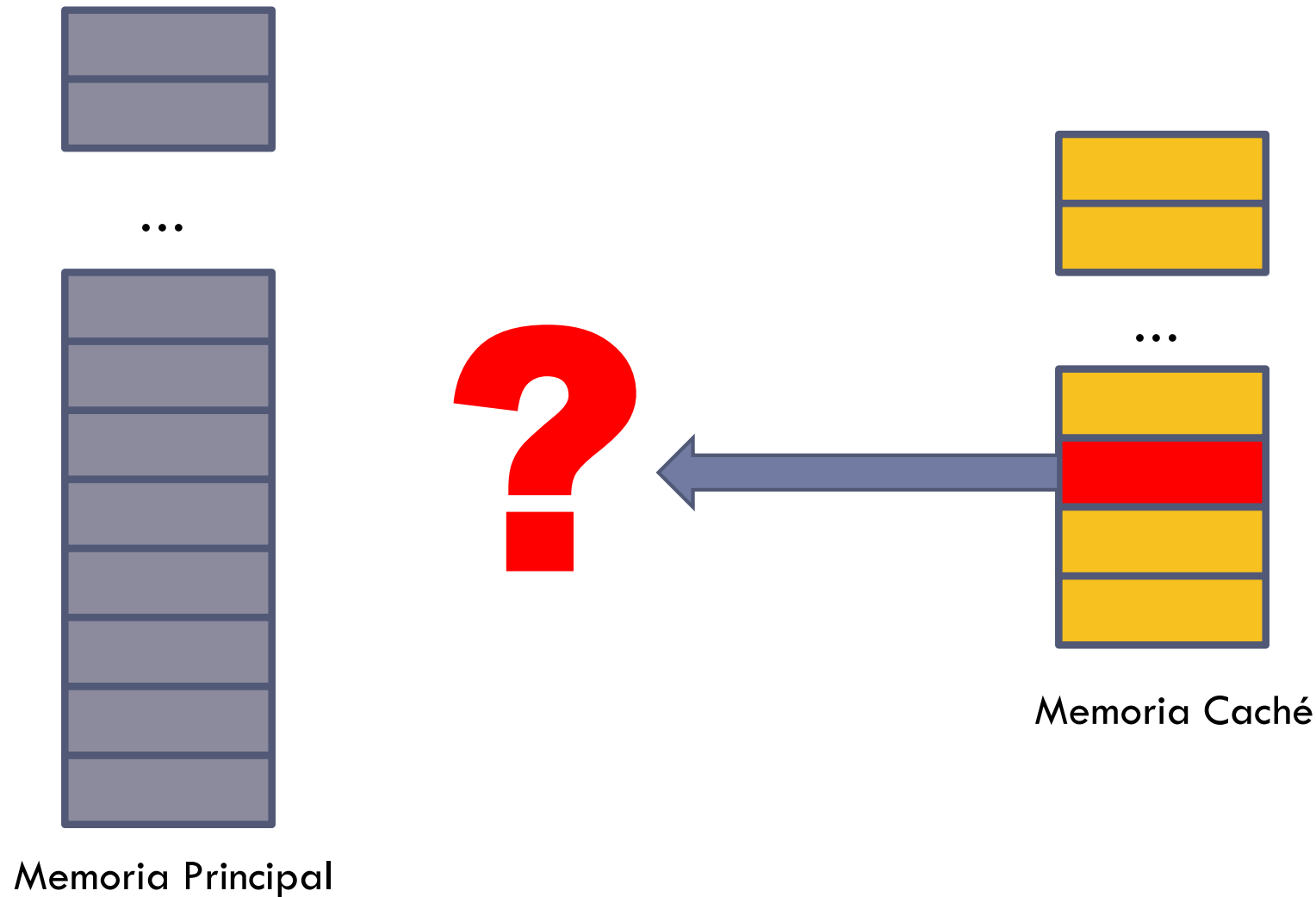
- ▶ Algoritmo que determina en qué lugares de la memoria caché se puede almacenar un bloque concreto de la memoria principal
- ▶ Un mecanismo que permita saber qué bloque concreto de memoria principal está en una línea de la memoria caché
  - ▶ Se asocian a las líneas etiquetas

# Ubicación en caché

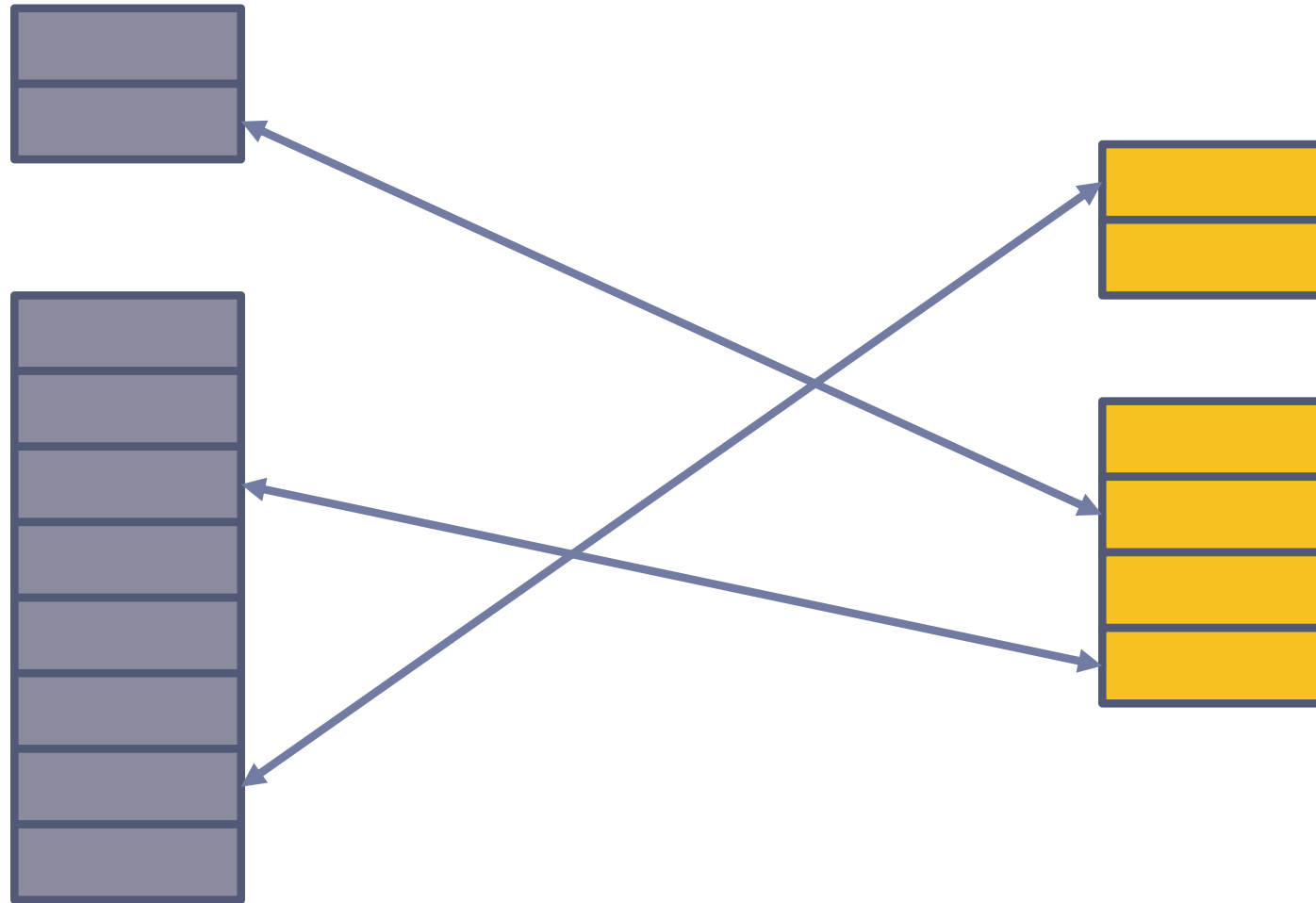




# Ubicación en memoria principal



# Función de correspondencia



¿Dónde están en caché los datos de una posición de M.P.?

# Funciones de correspondencia

- ▶ Función de correspondencia directa
- ▶ Función de correspondencia asociativa
- ▶ Función de correspondencia asociativa por conjuntos

# Correspondencia directa

00		
01		
10		
11		

Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea

## Memoria principal

Palabra de 32 bits

[illegible]

# Correspondencia directa

Bloque 0 - línea 0

00		
01		
10		
11		

Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea

## Memoria principal

Palabra de 32 bits

[illegible]

# Correspondencia directa

Bloque 1 - línea 1

00		
01		
10		
11		

Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea

## Memoria principal

Palabra de 32 bits

[illegible]

# Correspondencia directa

Bloque 2 - línea 2

00		
01		
10		
11		

Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea

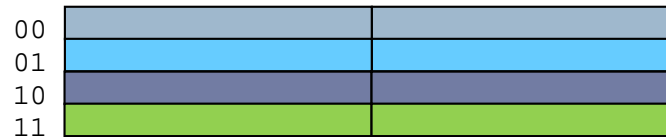
## Memoria principal

Palabra de 32 bits

[illegible]

# Correspondencia directa

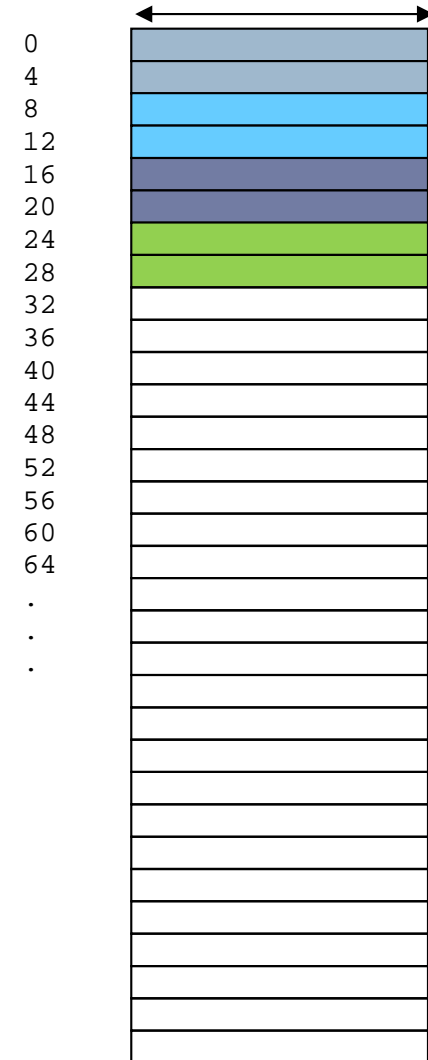
Bloque 3 - línea 3



Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea

## Memoria principal

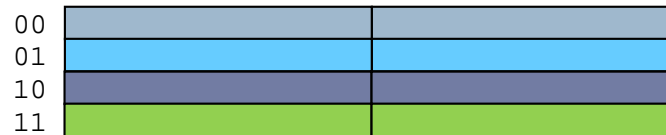
Palabra de 32 bits





# Correspondencia directa

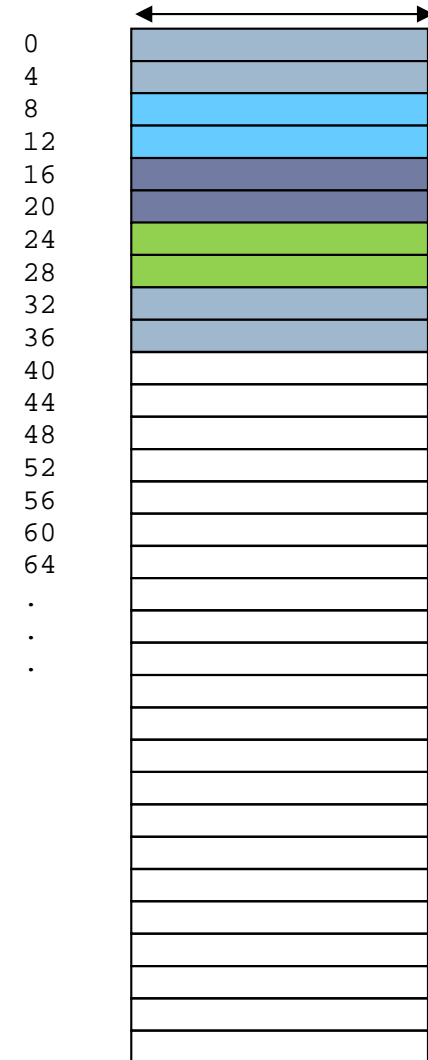
## Bloque 4 - línea 0



Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea

## Memoria principal

Palabra de 32 bits



# Correspondencia directa

- ▶ En general:

- ▶ El bloque de memoria  $K$  se almacena en la línea:

$$K \bmod \text{número de líneas}$$

# Correspondencia directa

00		
01		
10		
11		

Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea

Varios bloques en la misma línea

Memoria principal

Palabra de 32 bits

0000000	
0000100	
0001000	
0001100	
0010000	
0010100	
0011000	
0011100	
0100000	
0100100	
0101000	
0101100	
0110000	
0110100	
0111000	
0111100	
1000000	
1000100	
1001000	
1001100	
1010000	
1010100	
1011000	
1011100	
1100000	
1100100	
1101000	
1101100	
1110000	
1110100	
1111000	
1111100	

# Correspondencia directa

00		
01		
10		
11		

Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea

¿Cómo se sabe **qué** bloque de memoria se encuentra una determinada línea?

Ejemplo: la dirección 0100100

Memoria principal

Palabra de 32 bits

0000000	
0000100	
0001000	
0001100	
0010000	
0010100	
0011000	
0011100	
0100000	
0100100	
0101000	
0101100	
0110000	
0110100	
0111000	
0111100	
1000000	
1000100	
1001000	
1001100	
1010000	
1010100	
1011000	
1011100	
1100000	
1100100	
1101000	
1101100	
1110000	
1110100	
1111000	
1111100	

# Correspondencia directa

00		
01		
10		
11		

Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea

¿Cómo se sabe qué bloque de memoria se encuentra una determinada línea?

Ejemplo: la dirección 0100100

Se añade a cada línea una etiqueta

Memoria principal

Palabra de 32 bits

0000000	
0000100	
0001000	
0001100	
0010000	
0010100	
0011000	
0011100	
0100000	
0100100	
0101000	
0101100	
0110000	
0110100	
0111000	
0111100	
1000000	
1000100	
1001000	
1001100	
1010000	
1010100	
1011000	
1011100	
1100000	
1100100	
1101000	
1101100	
1110000	
1110100	
1111000	
1111100	

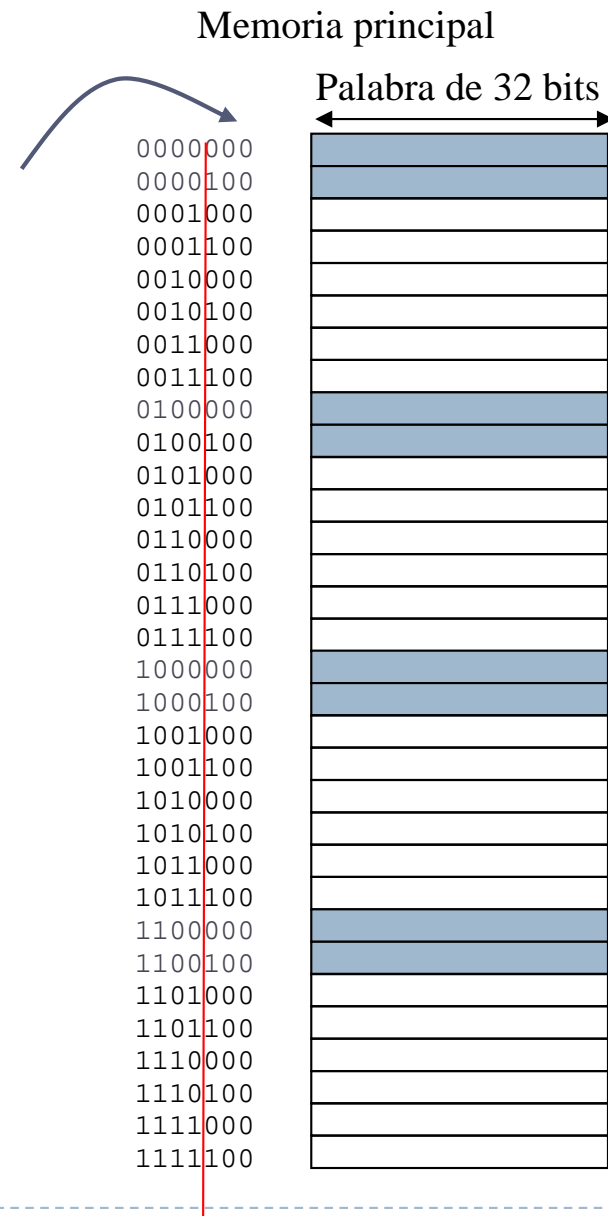
# Correspondencia directa

## Idea

¿qué byte dentro de la línea?  
Líneas de 8 bytes

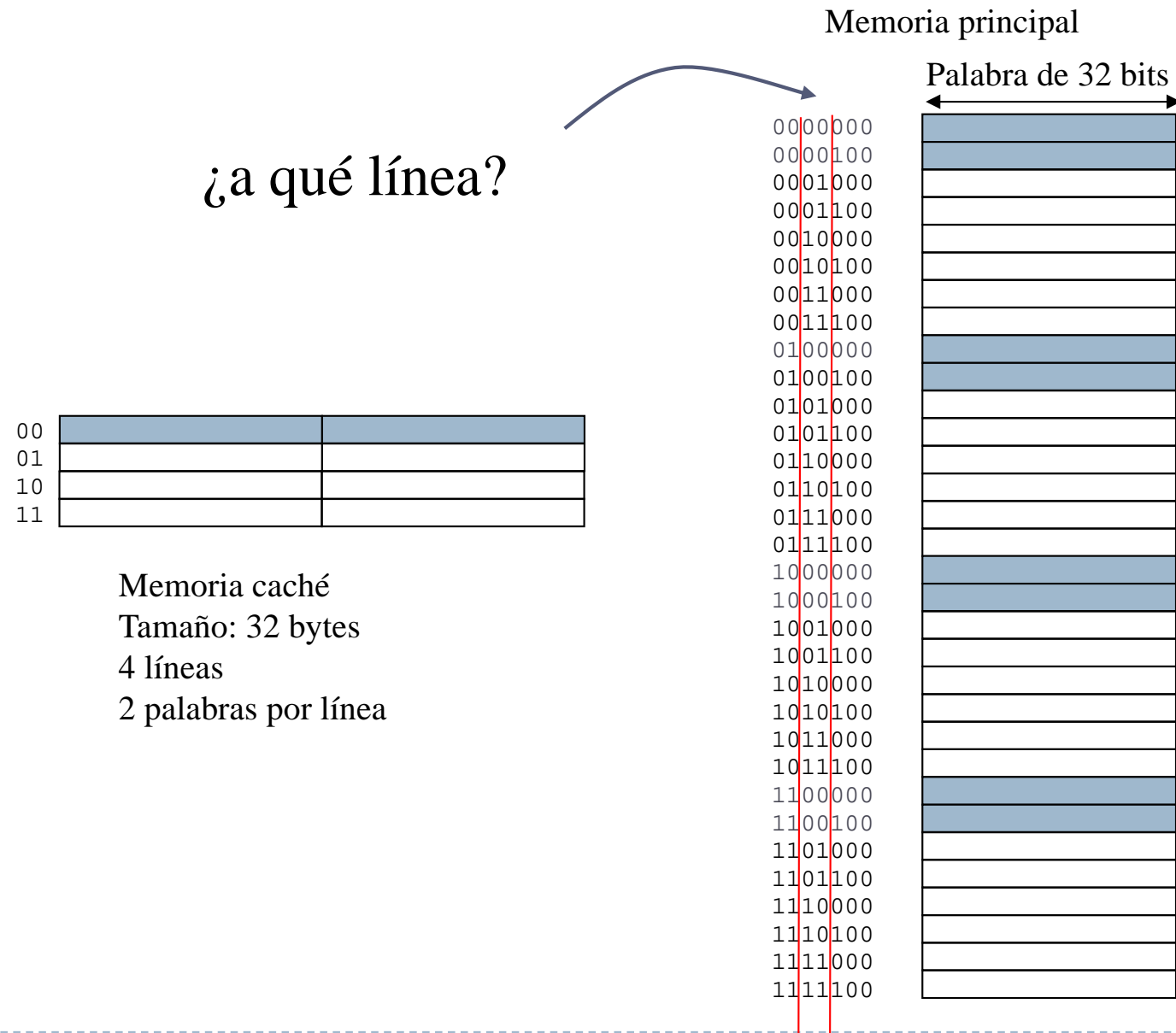
00		
01		
10		
11		

Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea



# Correspondencia directa

## Idea



# Correspondencia directa

## Idea

etiqueta asociada a la línea  
que diferencia los bloques  
que van a la misma línea

00			
01			
10			
11			

Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea

La etiqueta (bits superiores de la  
dirección) se almacena también  
en la memoria caché

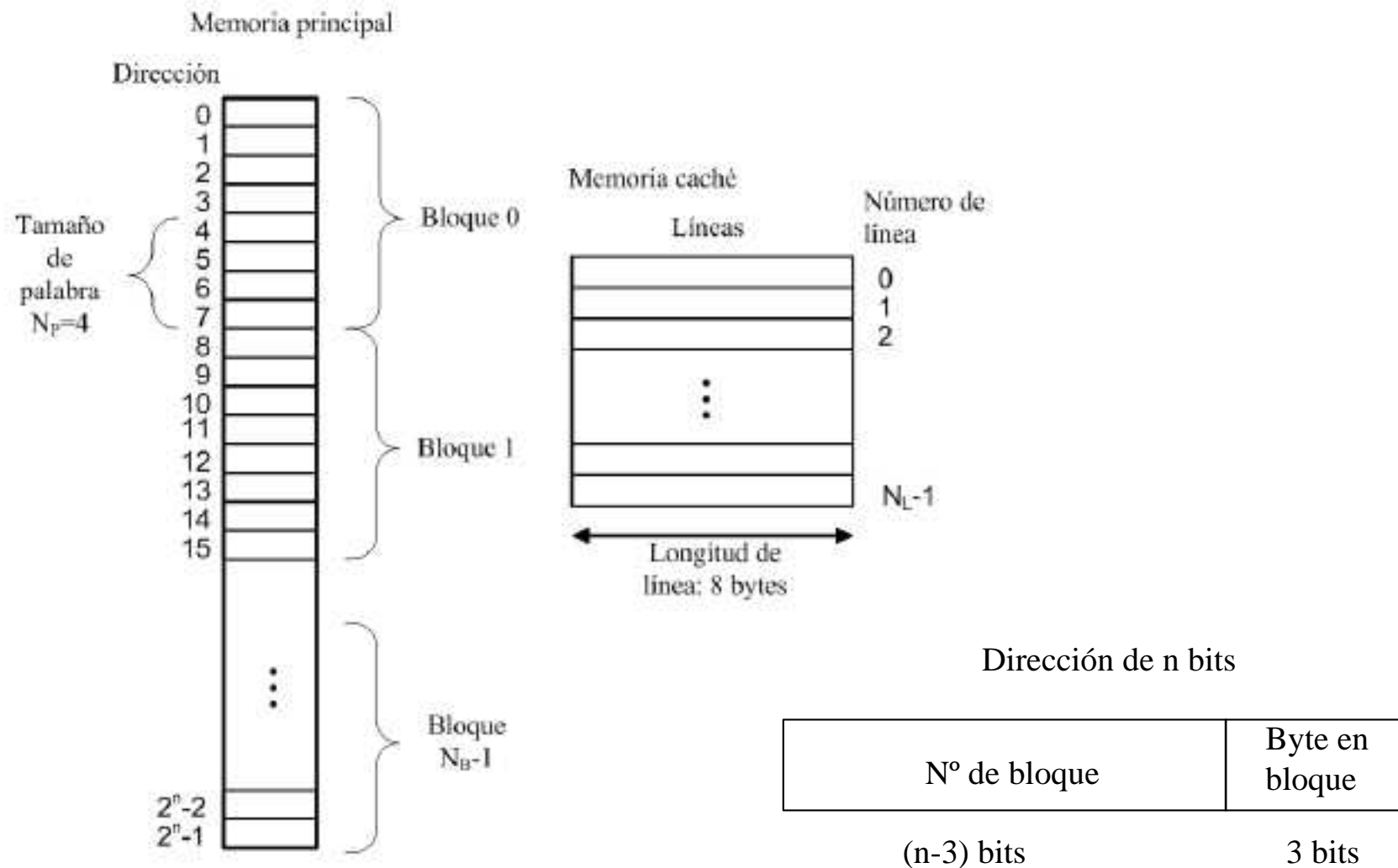
Memoria principal

Palabra de 32 bits

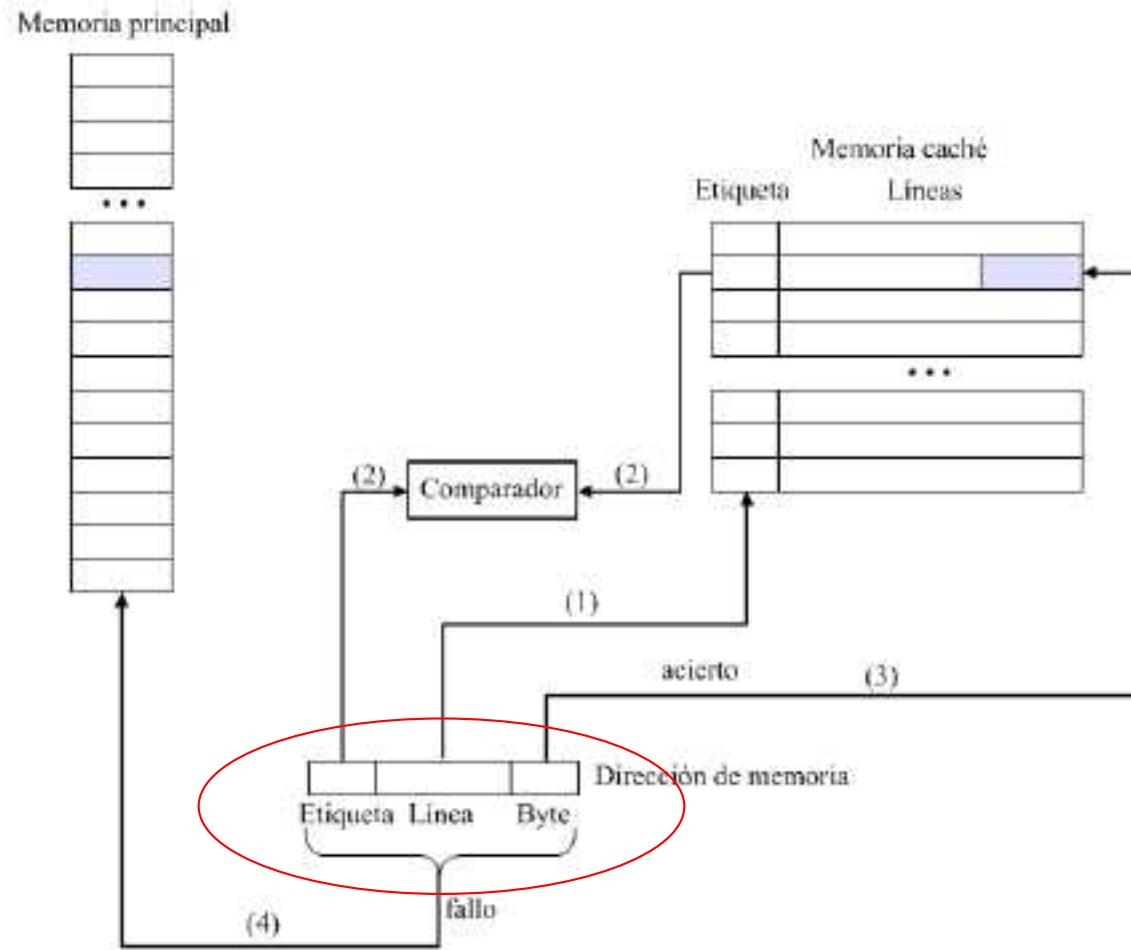
00	000000	
00	000100	
00	001000	
00	001100	
00	010000	
00	010100	
00	011000	
00	011100	
01	000000	
01	000100	
01	010000	
01	010100	
01	011000	
01	011100	
10	000000	
10	000100	
10	001000	
10	001100	
10	010000	
10	010100	
10	011000	
10	011100	
11	000000	
11	000100	
11	010000	
11	010100	
11	011000	
11	011100	



# Ejemplo de organización de la memoria caché



# Organización de una memoria caché con correspondencia directa

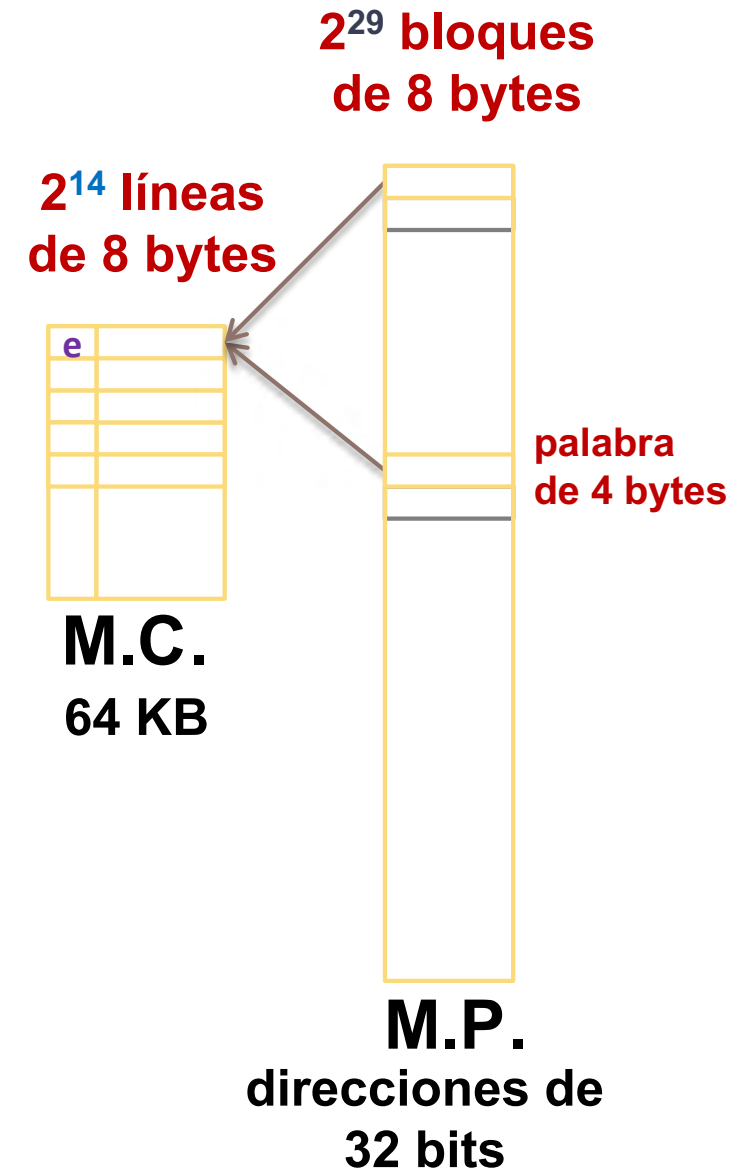


# Función de correspondencia directa Ejemplo

- ▶ Cada bloque de M.P. le corresponde una sola línea de caché (siempre la misma)
- ▶ La dirección de M.P. la determina:

32-17	14	3
etiqueta	línea	byte

- ▶ Si en 'línea' está 'etiqueta', entonces está el bloque en caché
- ▶ Simple, poco costosa, pero puede provocar muchos fallos dependiendo del patrón de accesos

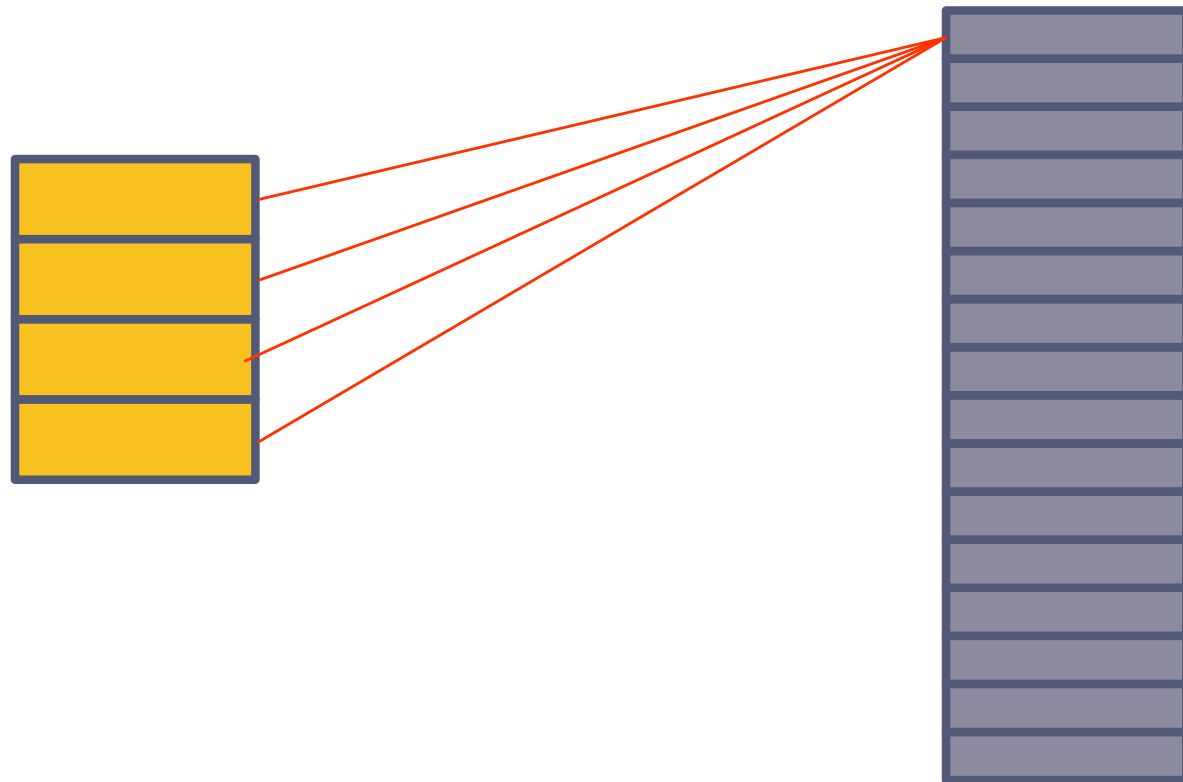


# Ejercicio

- ▶ Dado un computador de 32 bits con una memoria caché de 64 KB y bloques de 32 bytes. Si se utiliza correspondencia directa
  - ▶ ¿En qué línea de la memoria caché se almacena la palabra de la dirección 0x0000408A?
  - ▶ ¿Cómo se puede obtener rápidamente?
  - ▶ ¿En qué línea de la memoria caché se almacena la palabra de la dirección 0x1000408A?
  - ▶ ¿Cómo sabe la caché si la palabra almacenada en esa línea corresponde a la palabra de la dirección 0x0000408A o a la palabra de la dirección 0x1000408A?

# Correspondencia asociativa

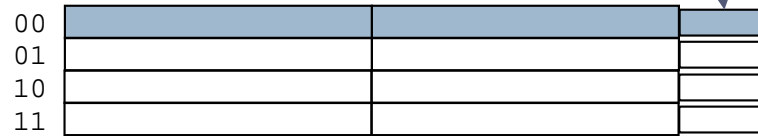
- ▶ Cada bloque de MP puede almacenarse en cualquier línea de la caché



# Correspondencia asociativa

## Idea

etiqueta asociada a la línea  
que diferencia los bloques  
que van a la misma línea

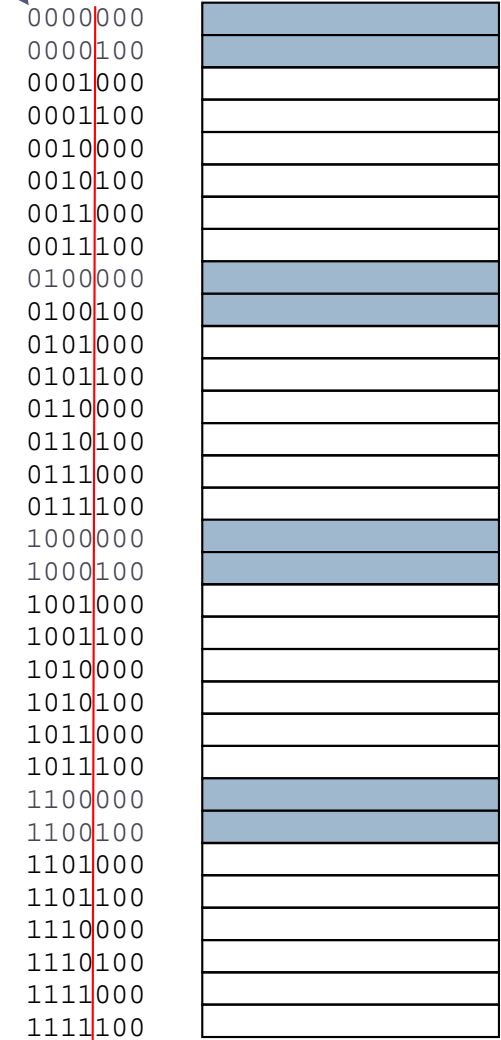


Memoria caché  
Tamaño: 32 bytes  
4 líneas  
2 palabras por línea

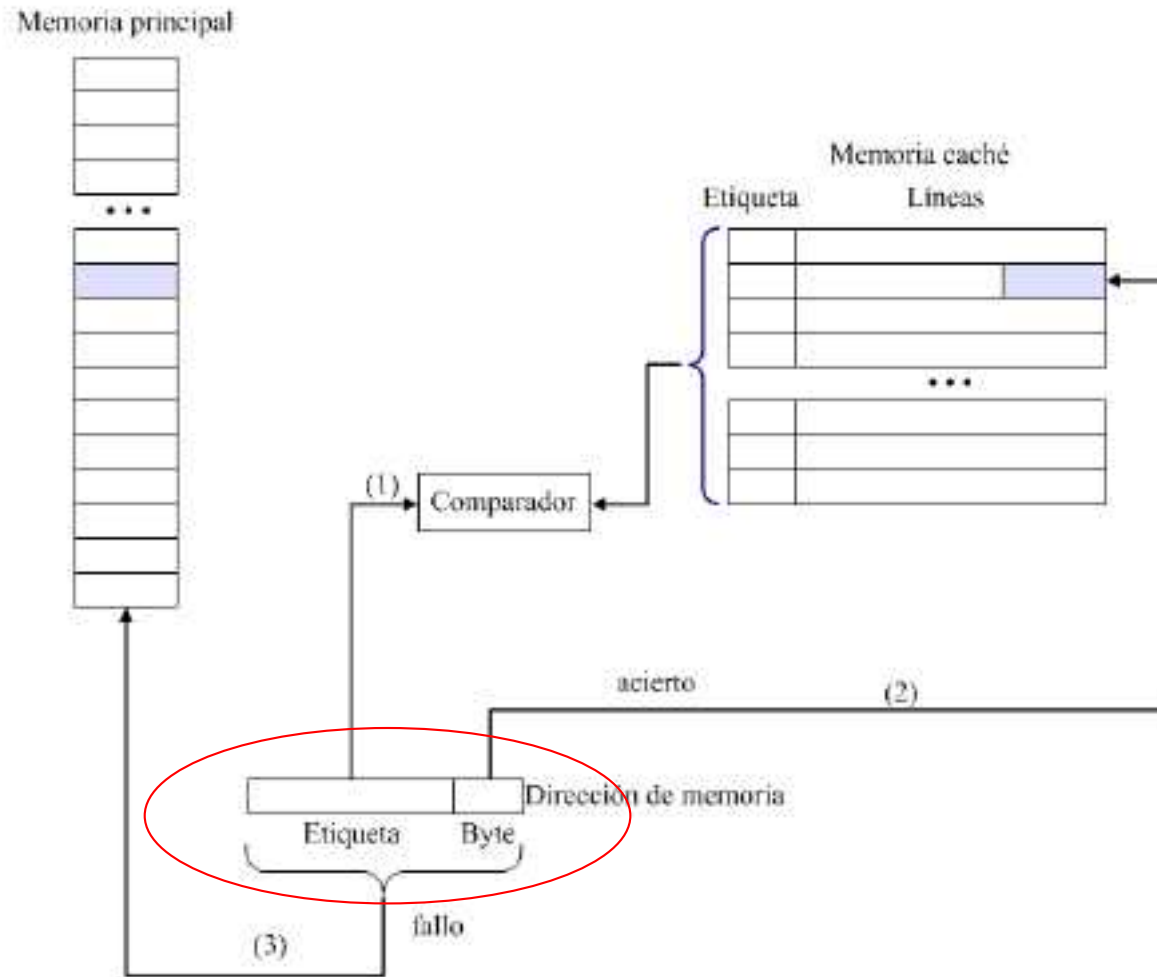
La etiqueta (bits superiores de la dirección) se almacena también en la memoria caché

## Memoria principal

Palabra de 32 bits



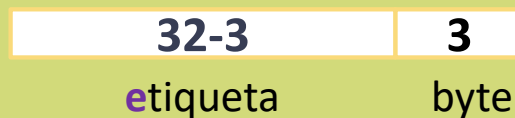
# Organización de una memoria caché con correspondencia asociativa



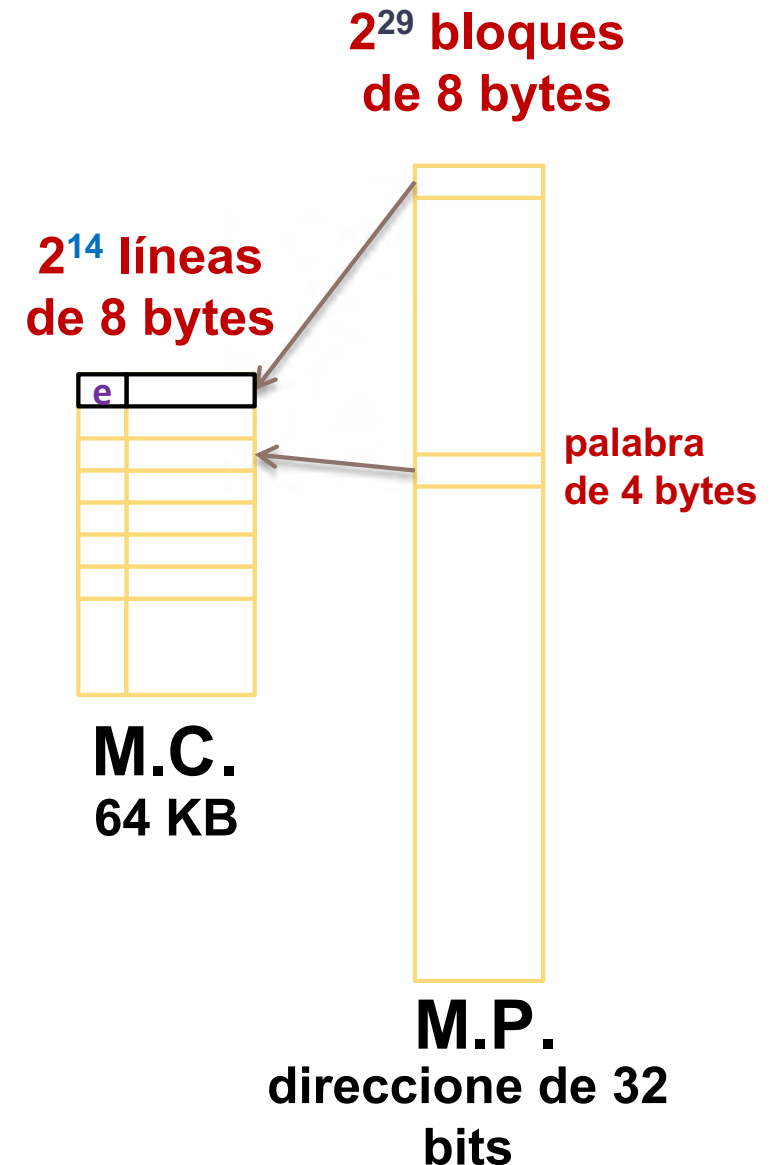
# Función de correspondencia asociativa

## Ejemplo

- ▶ Un bloque de M.P. puede cargarse en cualquier línea de caché
- ▶ La dir. de M.P. se interpreta como:



- ▶ Si hay un línea con 'etiqueta' en la caché, está allí el bloque
- ▶ Independiente del patrón de acceso, búsqueda costosa
- ▶ Etiquetas más grandes: cachés más grandes

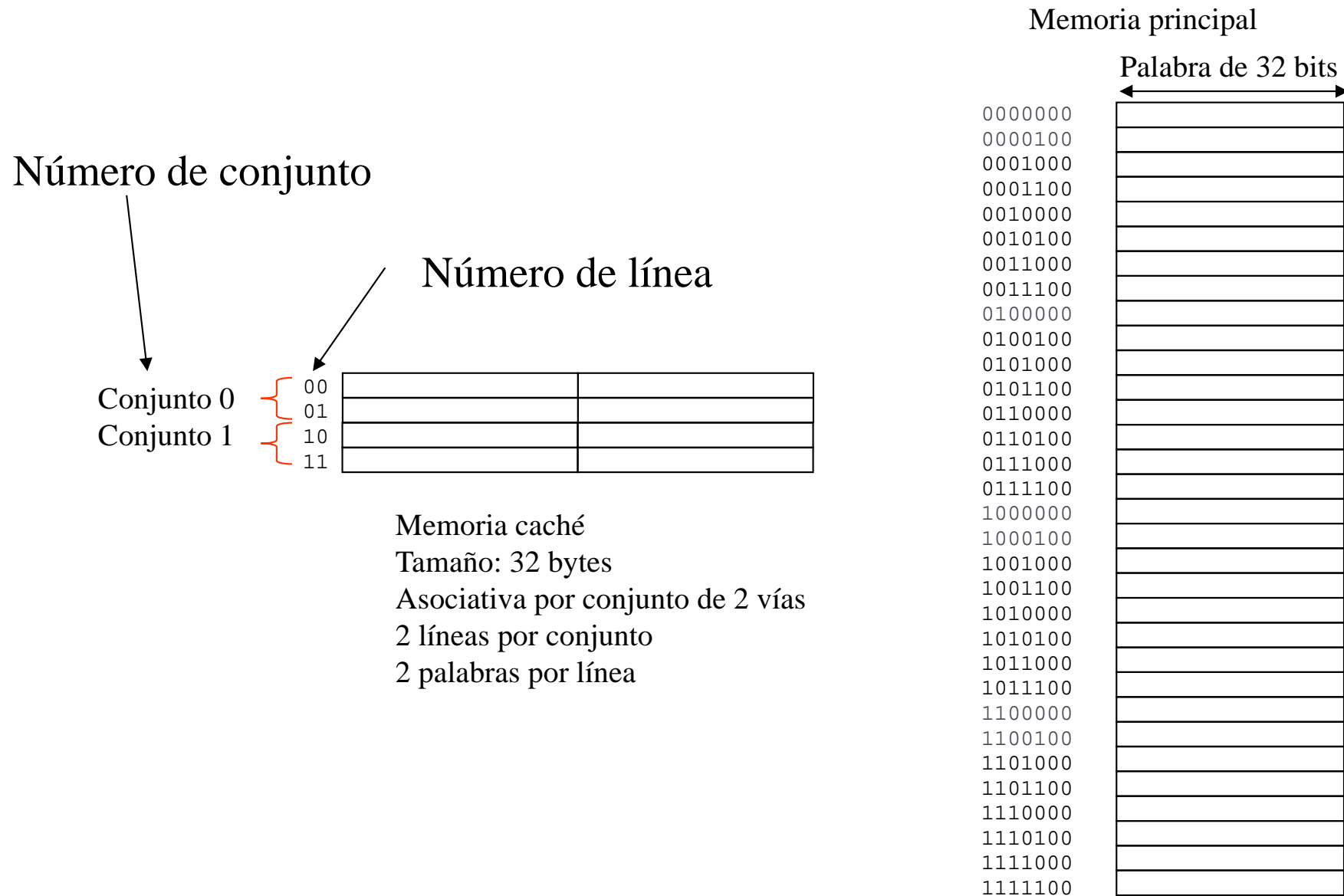




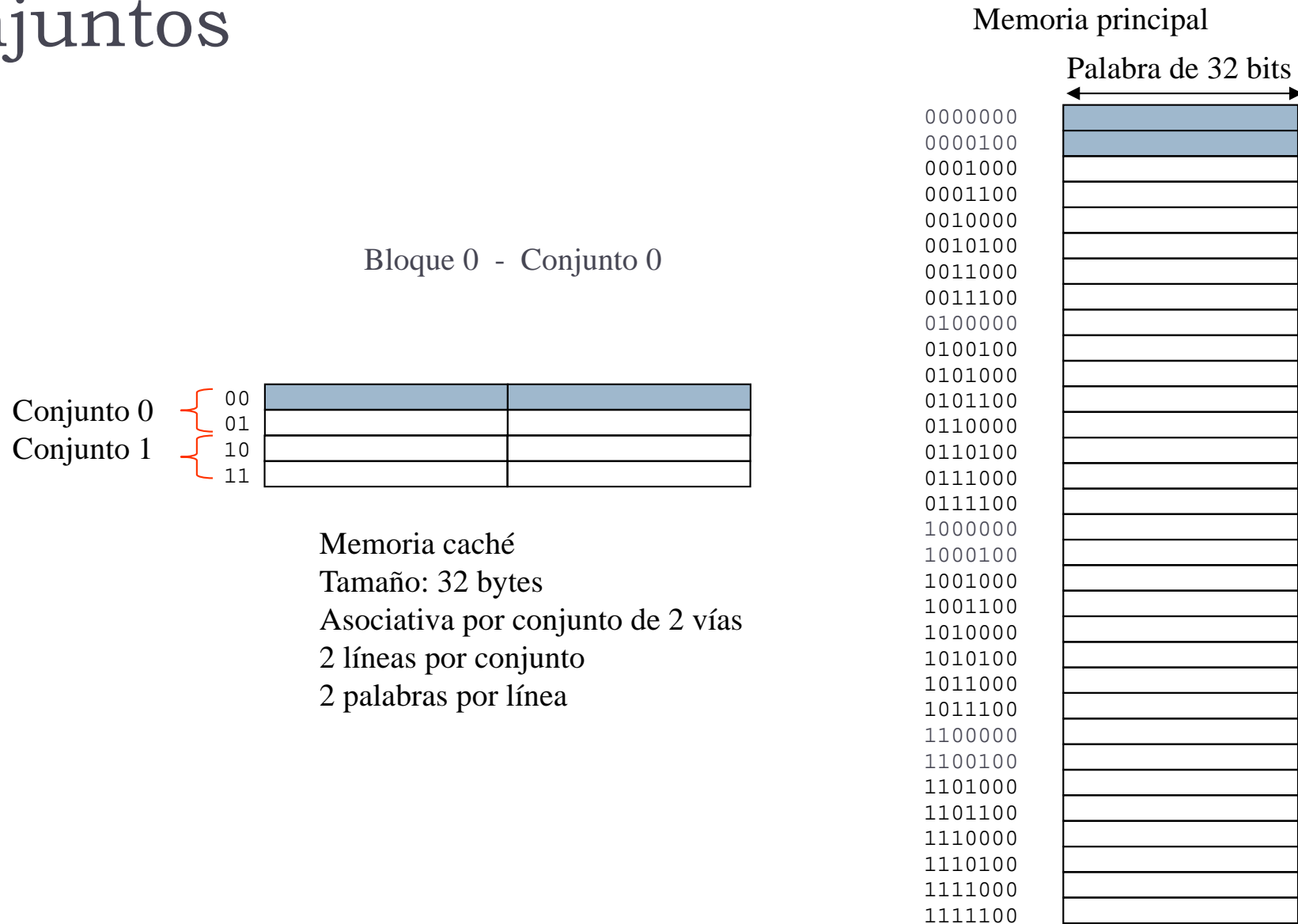
# Correspondencia asociativa por conjuntos

- ▶ La memoria se organiza en **conjuntos** de **líneas**
- ▶ Una memoria **caché asociativa por conjunto de K vías**:
  - ▶ Cada conjunto almacena K **líneas**
- ▶ Cada bloque siempre se almacena en el mismo conjunto
  - ▶ El bloque B se almacena en el conjunto:
    - ▶  **$B \bmod$**  número de conjuntos
- ▶ Dentro de un conjunto el bloque se puede almacenar en cualquiera de las **líneas** de ese conjunto

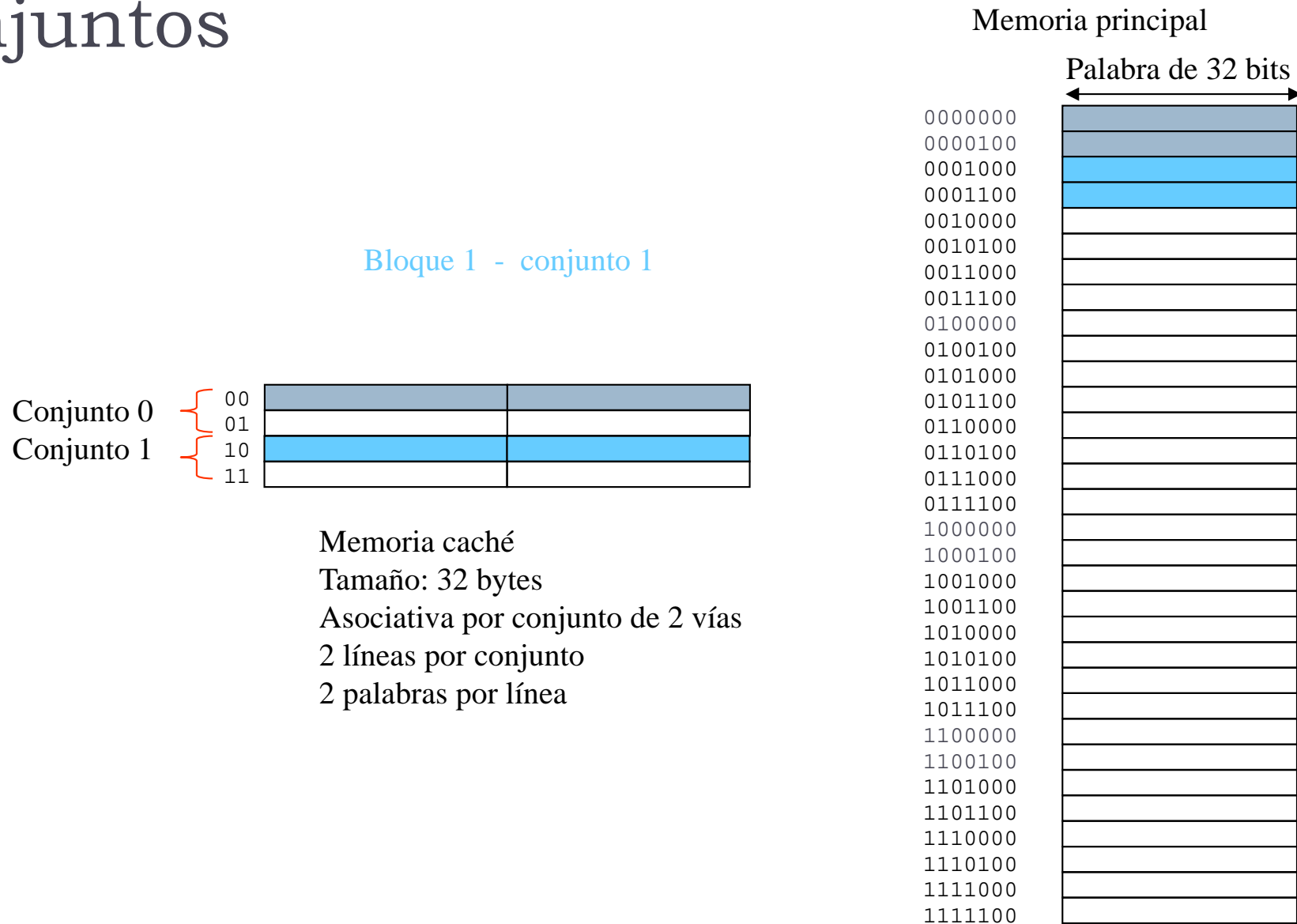
# Correspondencia asociativa por conjuntos



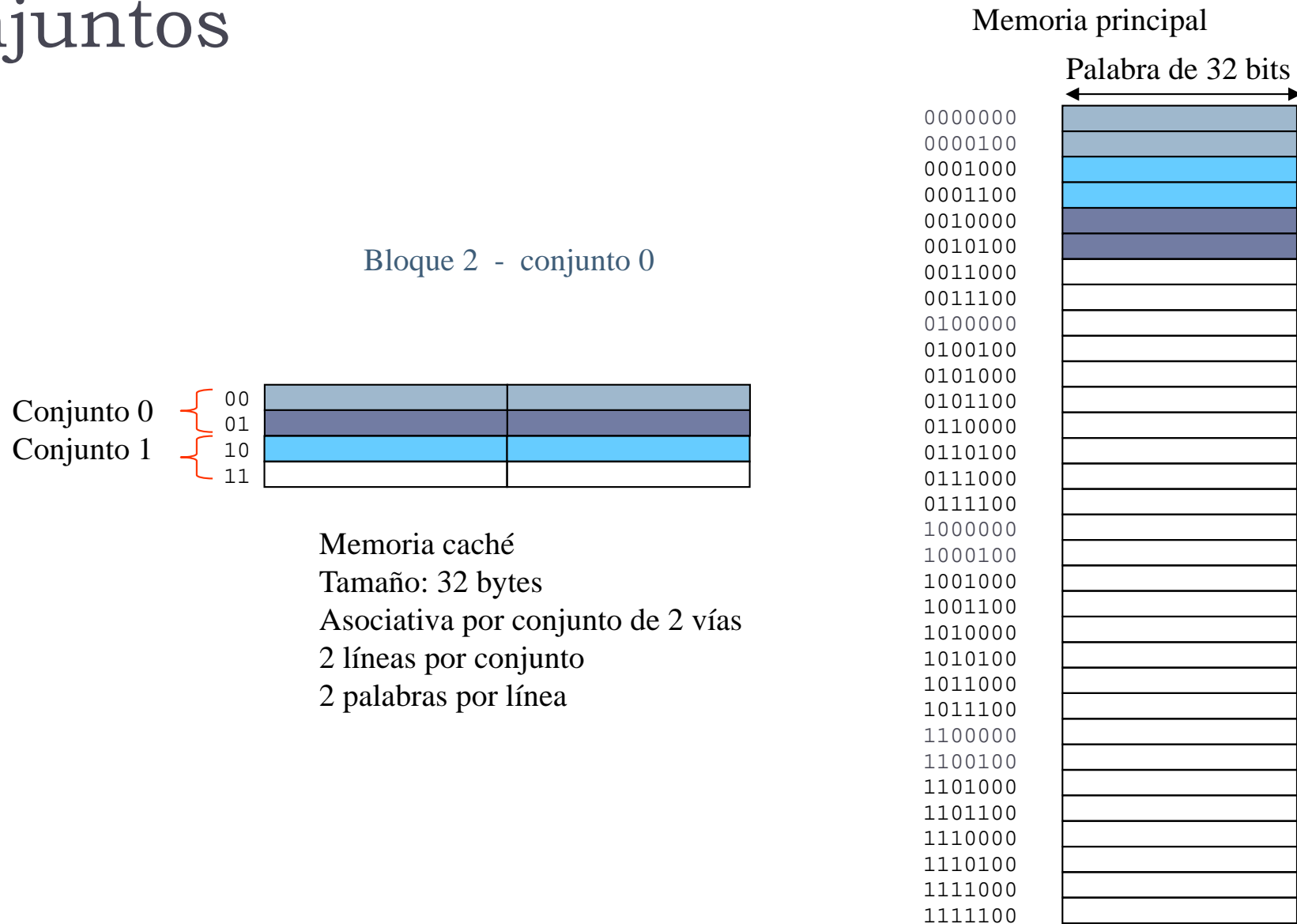
# Correspondencia asociativa por conjuntos



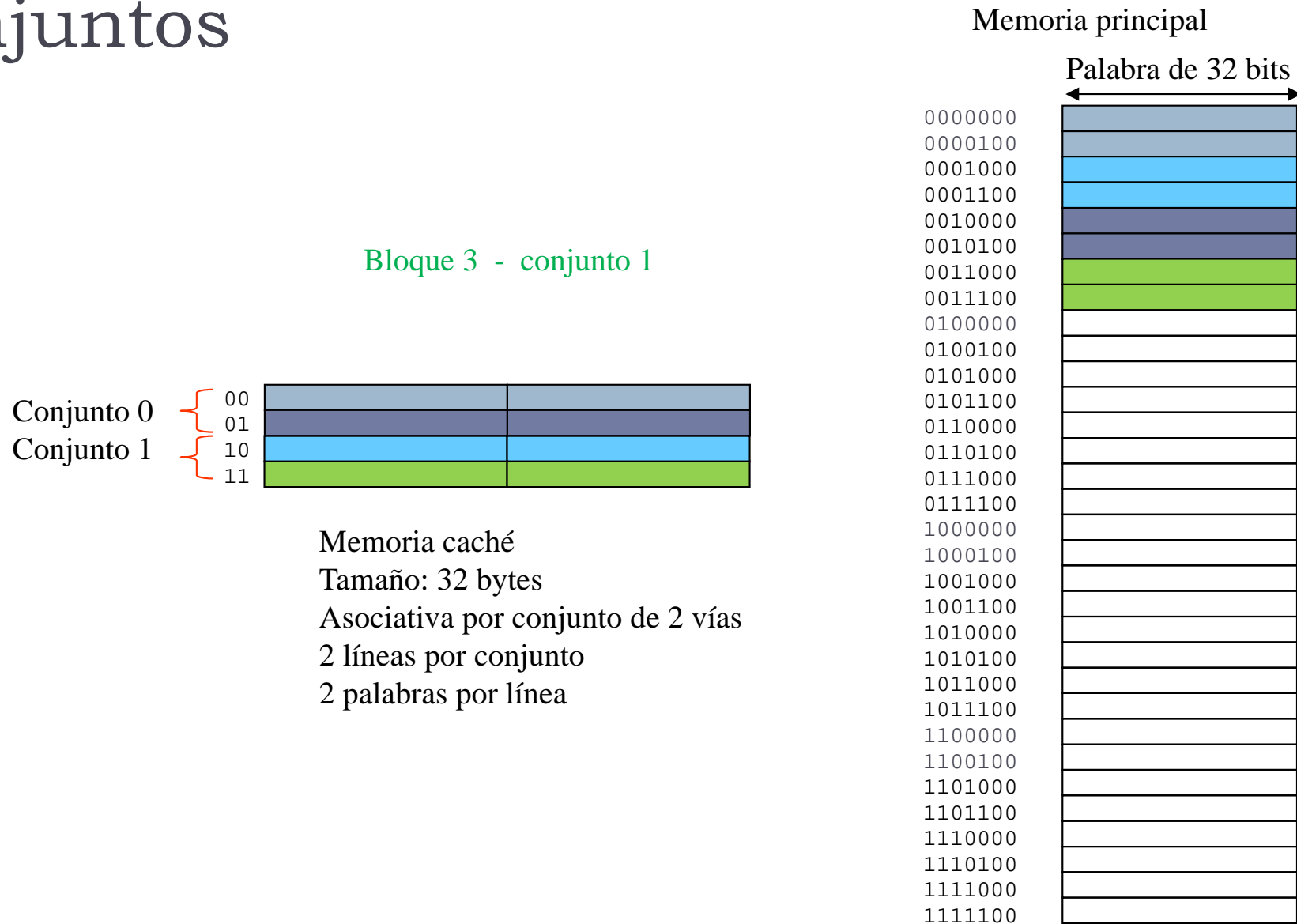
# Correspondencia asociativa por conjuntos



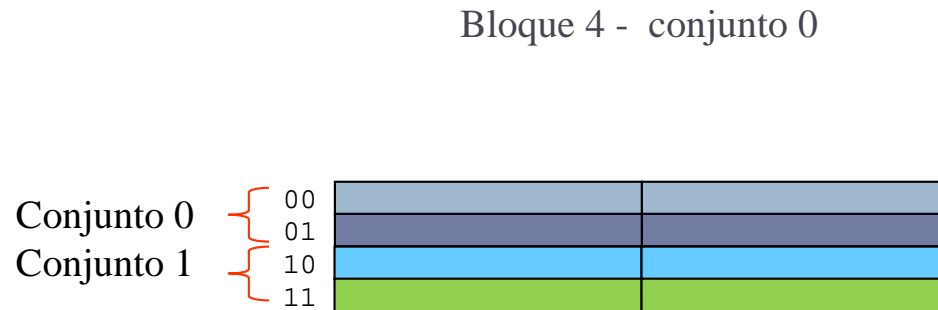
# Correspondencia asociativa por conjuntos



# Correspondencia asociativa por conjuntos

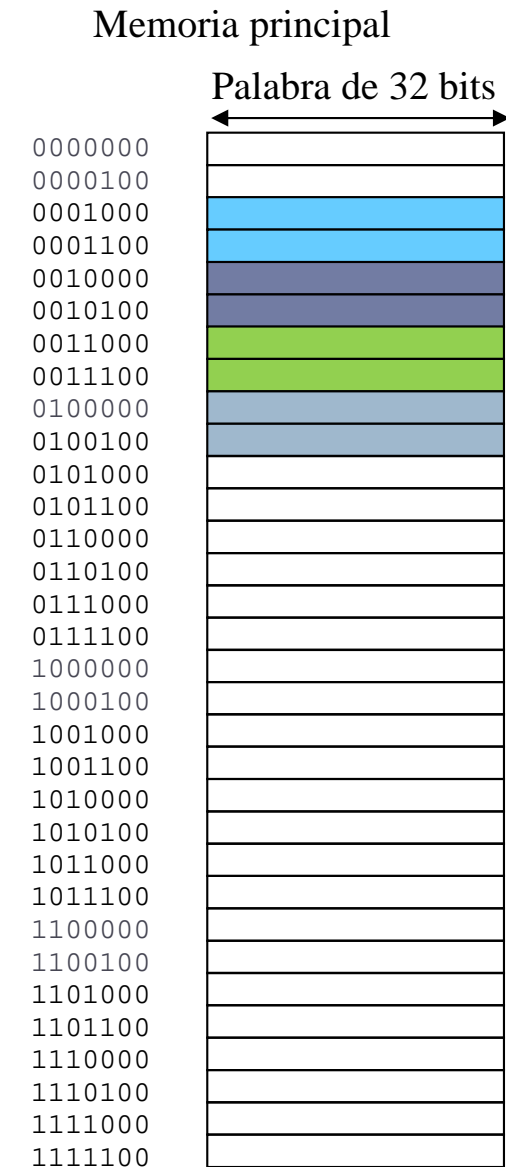


# Correspondencia asociativa por conjuntos



Memoria caché  
 Tamaño: 32 bytes  
 Asociativa por conjunto de 2 vías  
 2 líneas por conjunto  
 2 palabras por línea

Habría que eliminar la línea que estaba antes



# Correspondencia asociativa por conjuntos

¿qué byte dentro de la línea?  
Líneas de 8 bytes

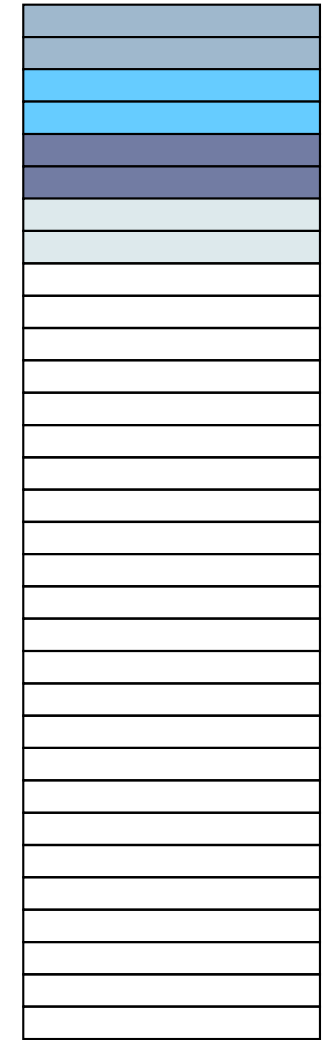


Memoria caché  
Tamaño: 32 bytes  
Asociativa por conjunto de 2 vías  
2 líneas por conjunto  
2 palabras por línea

Memoria principal

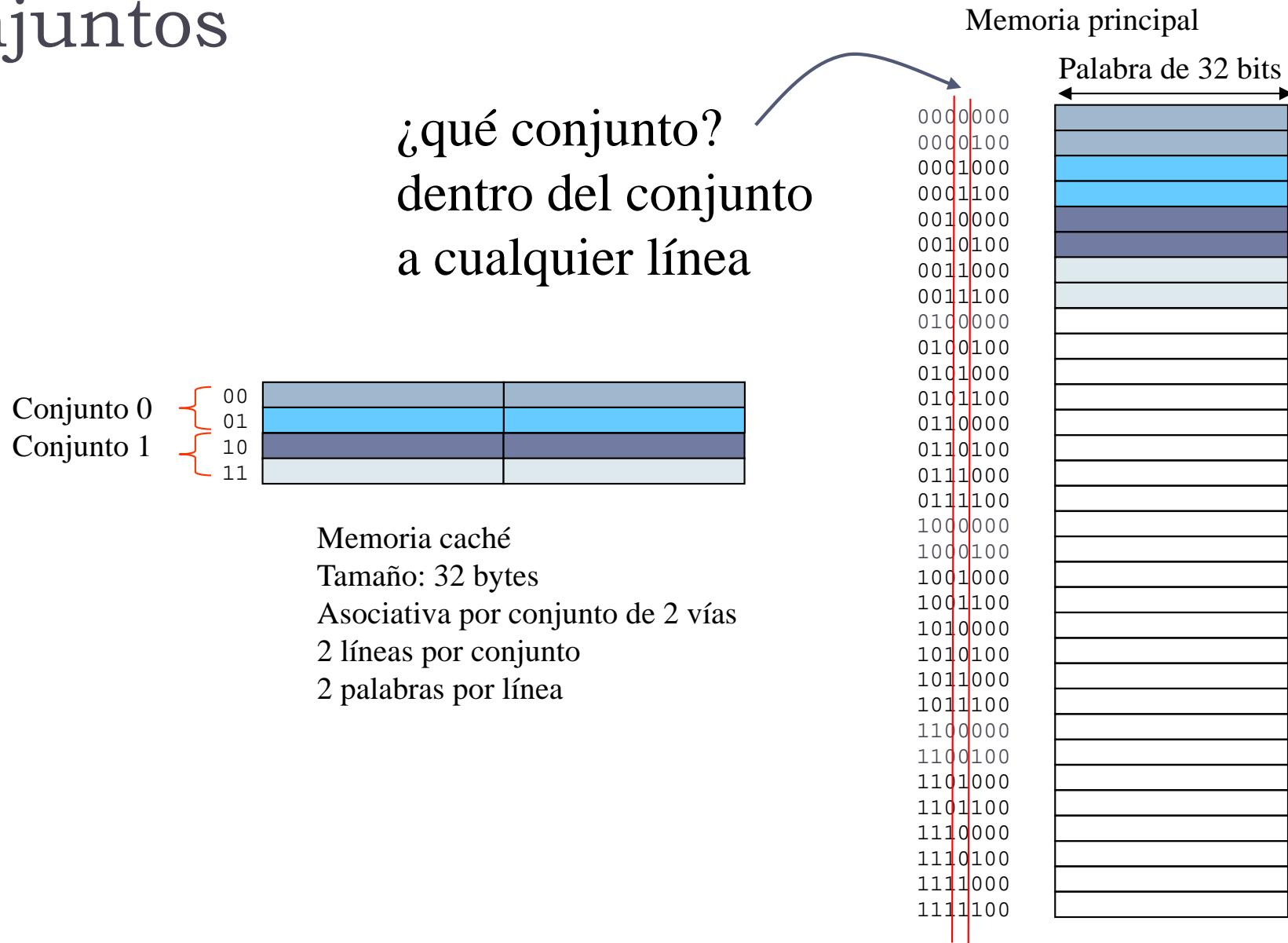
Palabra de 32 bits

00000000  
00001000  
00010000  
00011000  
00100000  
00101000  
00110000  
00111000  
01000000  
01001000  
01010000  
01011000  
01100000  
01101000  
01110000  
01111000  
10000000  
10001000  
10010000  
10011000  
10100000  
10101000  
10110000  
10111000  
11000000  
11001000  
11010000  
11011000  
11100000  
11101000  
11110000  
11111000

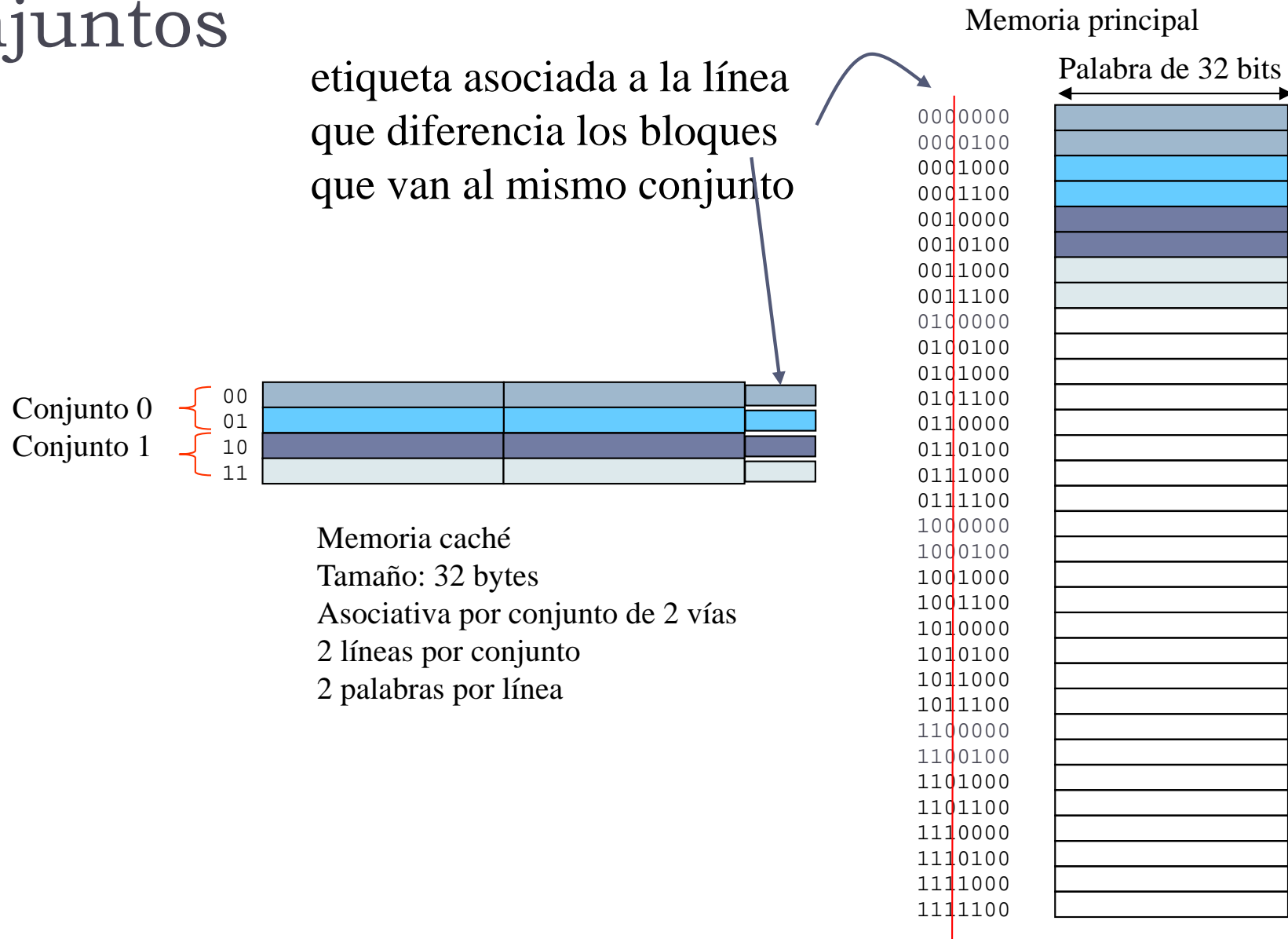




# Correspondencia asociativa por conjuntos



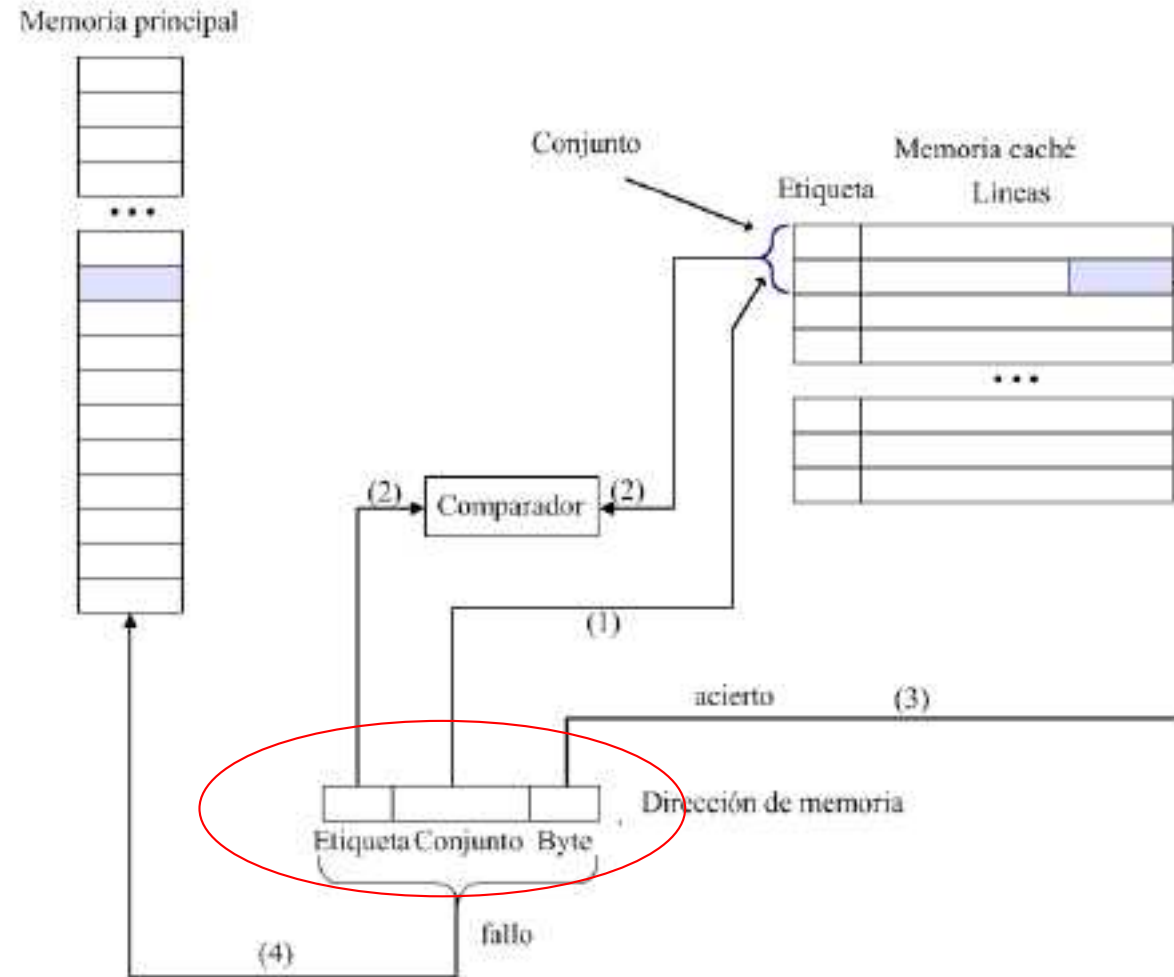
# Correspondencia asociativa por conjuntos



# Correspondencia asociativa por conjuntos

- ▶ Establece un compromiso entre flexibilidad y coste.
  - ▶ Es más flexible que la correspondencia directa.
  - ▶ Es menos costosa que la correspondencia asociativa.

# Organización de una memoria caché asociativa por conjuntos



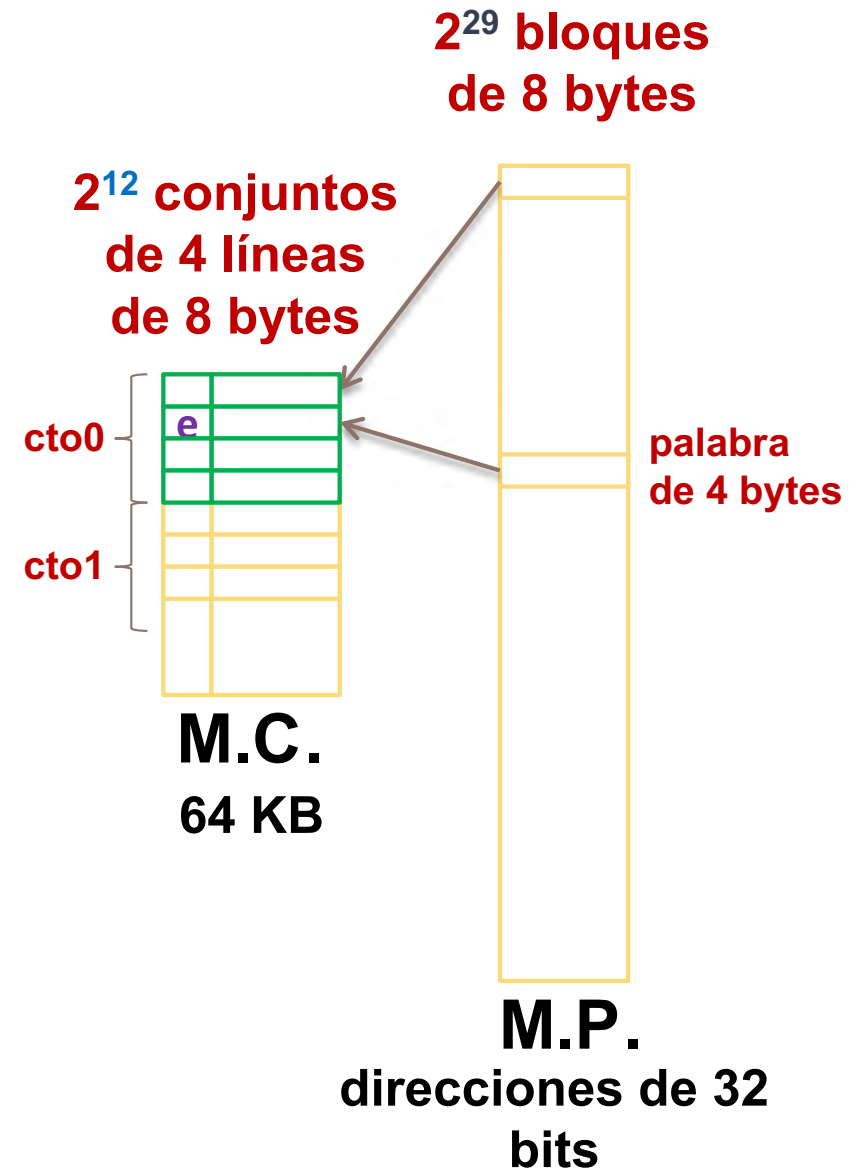
# Función de correspondencia asociativa por conjuntos. Ejemplo

## ► Asociativa por conjuntos:

- Un bloque de M.P. puede cargarse en cualquier línea de caché (va) de un conjunto determinado
- La dir. de M.P. se interpreta como:



- Si hay un línea con 'etiqueta' en el conjunto 'conjunto', está allí el bloque en caché
- [V] lo mejor de directa y asociativa  
[I] búsqueda menos costosa



# Sustitución de bloques

- ▶ Cuando todas las entradas de la caché contienen bloques de memoria principal:
  - ▶ Hace falta seleccionar una línea que hay que dejar libre para traer un bloque de la MP.
    - ▶ Directa: no hay posible elección
    - ▶ Asociativa: seleccionar una línea de la caché.
    - ▶ Asociativa por conjuntos: seleccionar una línea del conjunto seleccionado.
  - ▶ Existen diversos algoritmos para seleccionar la línea de la caché que hay que liberar.

# Algoritmos de sustitución

## ▶ FIFO

- ▶ *First-in-first-out*
- ▶ Sustituye la línea que lleva más tiempo en la caché.

## ▶ LRU:

- ▶ *Least Recently Used*
- ▶ Sustituye la línea que lleva más tiempo sin usarse.

## ▶ LFU:

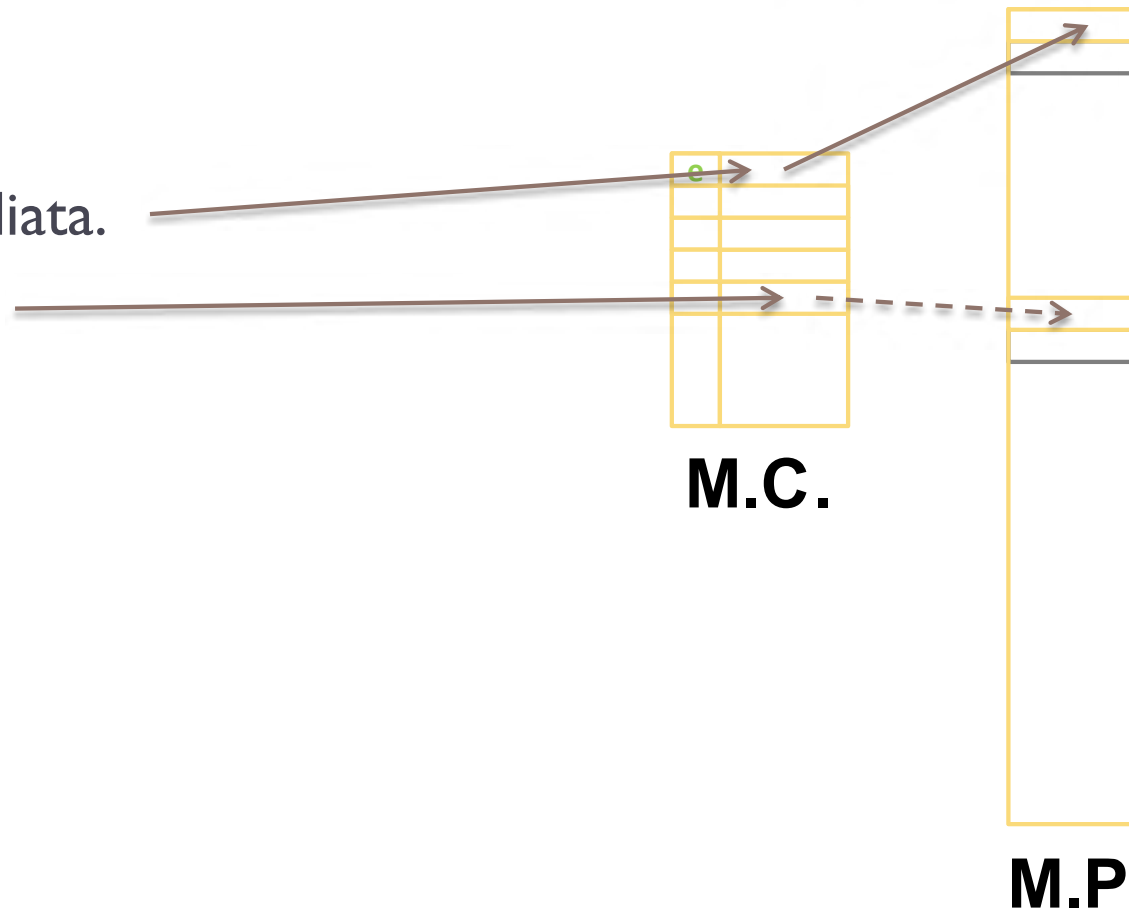
- ▶ *Least Frequently Used*
- ▶ Sustituye la línea que se ha usado menos veces.

# Políticas de escritura

- ▶ Cuando se modifica un dato en memoria caché, hay que actualizar en algún momento la memoria principal

- ▶ Alternativas:

- ▶ Escritura inmediata.
- ▶ Post-escritura.





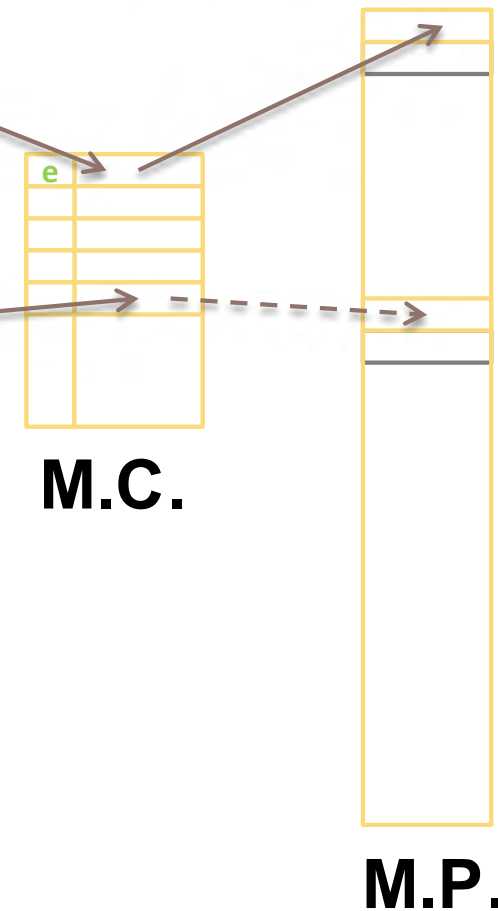
# Política de escritura

## ► Escritura **inmediata**:

- La escritura se hace tanto en M.P. como en cache
- [V] Coherencia
- [I] Mucho tráfico
- [I] Escrituras lentas

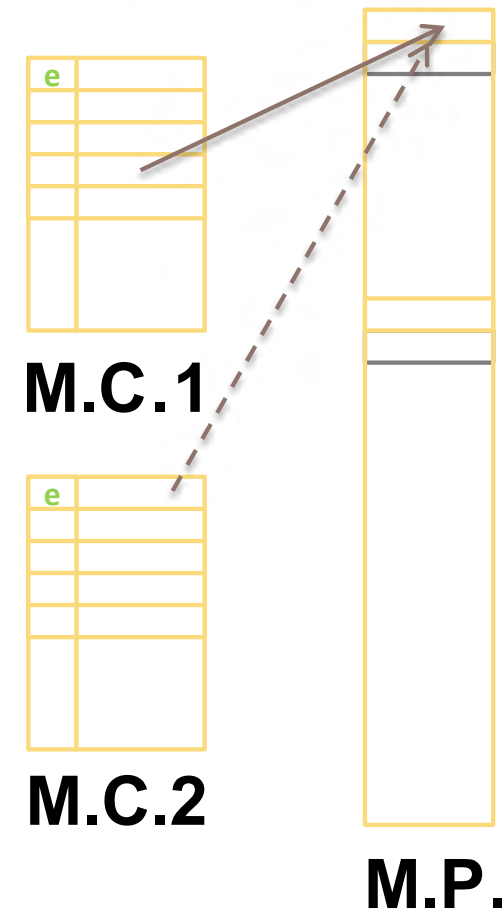
## ► Escritura **diferida**:

- La escritura solo se hace en la caché, indicando en un bit que no está volcada la línea en M.P.
- Al sustituir el bloque (o cuando ↓ tráfico con M.P.) se escribe en M.P.
- [V] Velocidad
- [I] Coherencia + inconsistencia

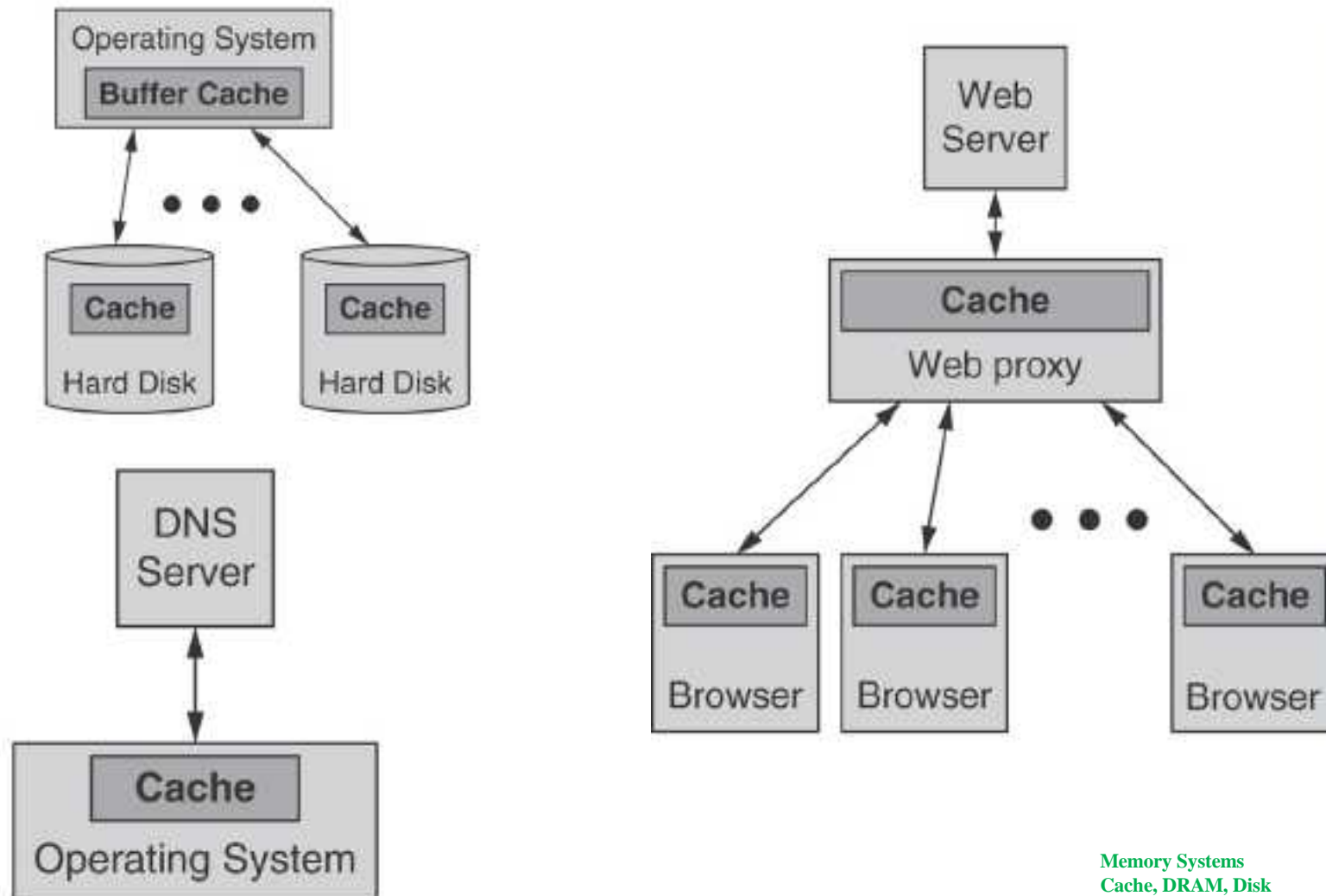


# Política de escritura

- ▶ Ej: CPU multicore con caché por core
  - ▶ Las escrituras en caché solo son vistas por un core
  - ▶ Si cada core escribe sobre una misma palabra, ¿cuál es el resultado final?
- ▶ Ej: módulo de E/S con acceso directo a M.P.
  - ▶ Actualización por DMA puede no ser coherente
- ▶ Porcentaje de referencias a memoria para escritura del orden del 15%.



# Ejemplos de cachés en otros sistemas



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

# Ejercicio

- ▶ Considere un computador de 32 bits con las siguientes características:
  - Memoria física instalada de 256 MB con un tiempo de acceso de 70 ns.
  - Direccionamiento de la memoria por bytes.
  - Tamaño de la memoria caché de 64 KB.
  - Tamaño de la línea 64 bytes.
  - La caché es asociativa por conjuntos de 4 vías.
  - El tiempo de acceso a la caché es de 5 ns y el tiempo de penalización en caso de fallo es de 100 ns.

# Ejercicio

## ► Se pide:

- a) ¿Cuántos bloques tiene la memoria principal?
- b) ¿Cuántos conjuntos tiene la memoria caché?
- c) Dada una dirección de memoria, indique qué partes de la dirección se utilizan para identificar la etiqueta, el conjunto y el byte dentro de la línea.  
Indique también el número de bits de cada parte.
- d) Dada la siguiente dirección de memoria  
0000 0011 1100 0011 0000 0000 1111 1000.  
En caso de encontrarse en la memoria caché  
¿en qué conjunto se almacenará?
- e) Si el tiempo medio de acceso al sistema de memoria es de 8 ns  
¿cuál es tasa de acierto necesaria para lograr este tiempo?

# Ejercicio (solución)

- a) La memoria tiene un tamaño de línea de 64 bytes =  $2^6$  bytes.  
Por tanto, el número de bloques de memoria principal será

$$\begin{aligned}\text{nbloques} &= \text{tamaño memoria} / \text{tamaño de línea} = \\ &256 \text{ MB} / 64 = \\ &256 \times 2^{20} / 64 = 256 * 2^{14} \text{ bloques}\end{aligned}$$

- b) El número de líneas de memoria caché es

$$\begin{aligned}\text{nlineas} &= \text{tamaño memoria} / \text{tamaño de línea} = \\ &64 \text{ KB} / 64 \text{ bytes} = \\ &2^{16} / 2^6 = 2^{10} = 1024 \text{ líneas}\end{aligned}$$

$$\text{Conjuntos} = \text{N}^\circ \text{ líneas} / \text{N}^\circ \text{ vías} = 1024 / 4 = 256 \text{ conjuntos.}$$

# Ejercicio (solución)

- c) La dirección de una caché asociativa por conjuntos se divide en tres partes: etiqueta, conjunto y byte dentro de la línea.
- ▶ Byte: el tamaño de la línea es 64 bytes =  $2^6$  bytes.  
Se necesitan, por tanto 6 bits para identificar el byte dentro de la línea.
  - ▶ Conjunto: Hay 256 conjuntos =  $2^8$ ,  
por lo que se necesitan 8 bits para identificar un conjunto
  - ▶ Etiqueta: para la etiqueta se emplean el resto de los bits de la dirección =  
 $32 - 6 - 8 = 18$

La dirección quedará:

Etiqueta (18 bits)	Conjunto (8 bits)	Byte (6 bits)
--------------------	-------------------	---------------

# Ejercicio (solución)

- d) Utilizamos el formato de la dirección del apartado anterior:

Etiqueta (18 bits)	Conjunto (8 bits)	Byte (6 bits)
--------------------	-------------------	---------------

El byte asociado a esta dirección se encontrará en el **conjunto 3**

- e) El cálculo del tiempo medio de acceso a memoria se hace con la siguiente fórmula:

$$\triangleright T_{\text{medio}} = t_c + (1-h) * t_{\text{fallo}}$$

$$\triangleright 8 = 5 + (1-h) * 100$$

Despejando h, se tiene  $h = 97/100 = 0,97$  (tanto por uno)

Es decir, **una tasa de acierto del 97 %**



# Ejercicio

- ▶ Sea un computador con una memoria caché y principal con las siguientes características:
  - ▶ Tiempo de acceso a memoria caché de 4 ns
  - ▶ Tiempo de acceso a memoria principal de 80 ns
  - ▶ Tiempo para servir un fallo de caché de 120 ns
  - ▶ Política de escritura inmediata

En este computador se ha observado que la tasa de aciertos a la memoria caché es del 95 % y que cada 100 accesos, 90 son de lectura. Calcular el tiempo medio de acceso a memoria.

# Ejercicio

- ▶ Sea un computador dotado de una memoria cache con las siguientes características:
  - ▶ Tamaño: 16 KB con bloques de 32 bytes (8 palabras)
  - ▶ Tiempo de acceso: 10ns
  - ▶ Esta memoria está conectada a través de un bus de 32 bits a una memoria principal que es capaz de transferir un bloque de 8 palabras en 120 ns
  - ▶ Política de escritura: post-escritura o escritura diferida.
  - ▶ Se pide:
- ▶ Calcular la tasa de aciertos que es necesaria para que el tiempo medio de acceso al sistema de memoria sea de 20 ns.

# Ejercicio

- ▶ Se dispone de un computador con una memoria caché con un tamaño de 64 KB. El tamaño de la línea es de 64 bytes. La caché tiene un tiempo de acceso de 20 ns y un tiempo de penalización por fallo de 120 ns. La caché es asociativa por conjuntos de dos vías. Se pide:
  - ▶ Indique el número total de líneas de caché
  - ▶ Indique el número de conjuntos que tiene la caché.
  - ▶ Indique el número de líneas por conjunto
  - ▶ Haga un dibujo con la estructura de la caché
  - ▶ Diga cuánto tiempo tardaríamos en obtener un dato si se produce un fallo en la caché.

# Ejercicio

- ▶ Sea un computador de 32 bits con el juego de instrucciones del MIPS, que ejecuta el siguiente fragmento de código cargado a partir de la dirección 0x00000000

```
        li      $t0, 1000
        li      $t1, 0
        li      $t2, 0
bucle:  addi     $t1, $t1, 1
        addi     $t2, $t2, 4
        beq      $t1, $t0, bucle
```

- ▶ Este computador dispone de una memoria caché asociativa por conjunto de 4 vías, de 32 KB y líneas de 16 bytes. Calcule de forma razonada el número de fallos de caché y la tasa de aciertos que produce el fragmento de código anterior, asumiendo que se ejecuta sin ninguna interrupción y que la memoria caché está inicialmente vacía

# Ejercicio

Se dispone de un computador con direcciones de memoria de 32 bits, que direcciona la memoria por bytes. El computador dispone de una memoria caché asociativa por conjuntos de 4 vías, con un tamaño de línea de 64 bytes. Dicha caché tiene un tamaño de 128 KB. El tiempo de acceso a la memoria caché es de 2 ns y el tiempo necesario para tratar un fallo de caché es de 80 ns. Considere el siguiente fragmento de programa.

```
float v1[10000];  
float v2[10000];  
  
for (i = 0; i < 10000; i = i + 1)  
    v1[i] = v1[i] + v2[i];
```

Indique de forma razonada:

- a) El tamaño en MB de la memoria que se puede direccionar en este computador.
- b) El número de palabras que se pueden almacenar en la memoria caché de este computador.
- c) El número de líneas de la caché y número de conjuntos de la caché.
- d) Indique la tasa de aciertos necesaria para que el tiempo medio de acceso al sistema de memoria de este computador sea de 10 ns.
- e) Indique de forma razonada la tasa de aciertos a la caché para el fragmento de código anterior teniendo en cuenta solo los accesos a datos (considere que la variable *i* se almacena en un registro y que la caché está inicialmente vacía).