Grupo ARCOS

uc3m | Universidad Carlos III de Madrid

Tema 5 (II)
Jerarquía de Memoria

Estructura de Computadores
Grado en Ingeniería Informática

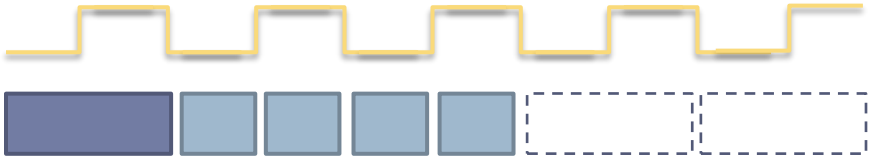
Contenidos

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

Característica de la memoria principal

- 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



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:

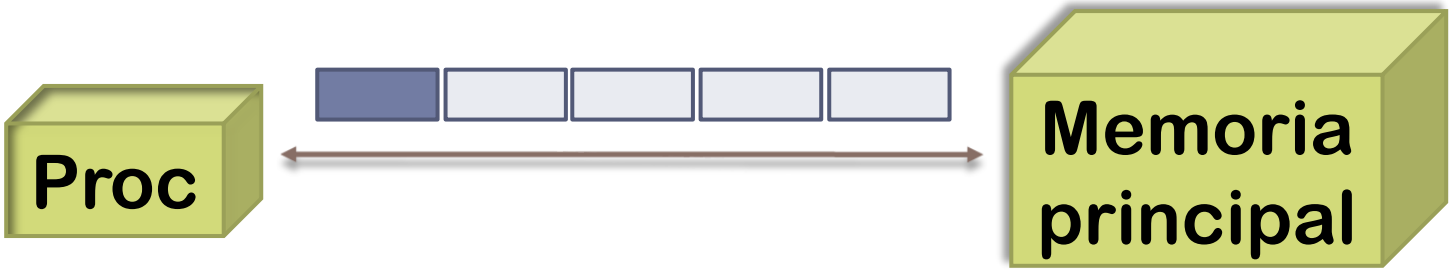
  - proximidad espacial
    - Secuencia de instrucciones
    - Acceso secuencial a arrays
  - proximidad temporal
    - Bucles

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

.text
 main:    li   t0 0
          la   t1 vector
          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
          j    b2
     fb2: jr ra
```

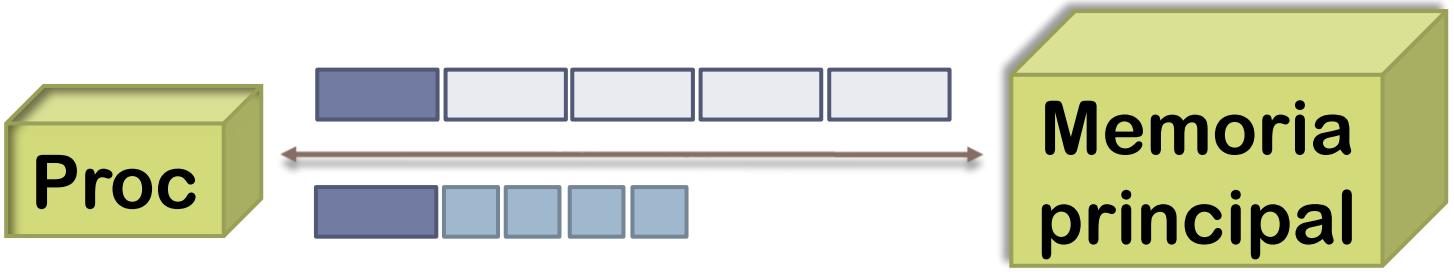
Objetivo de la memoria caché: aprovechar los accesos contiguos

- 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.



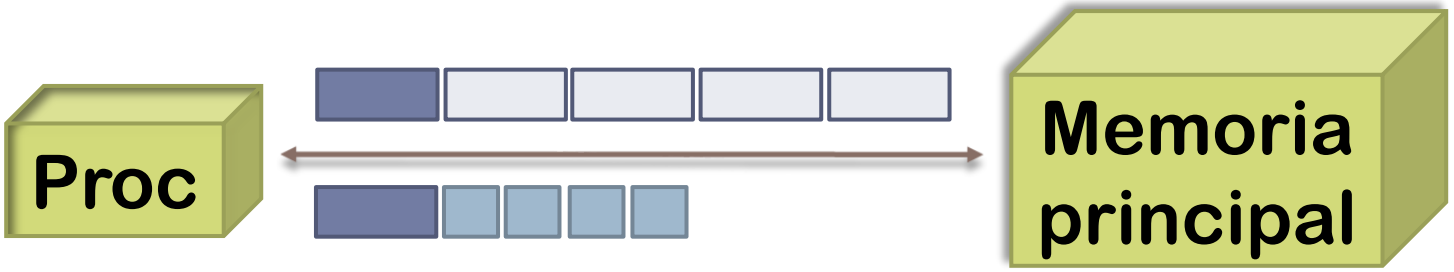
Objetivo de la memoria caché: aprovechar los accesos contiguos

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



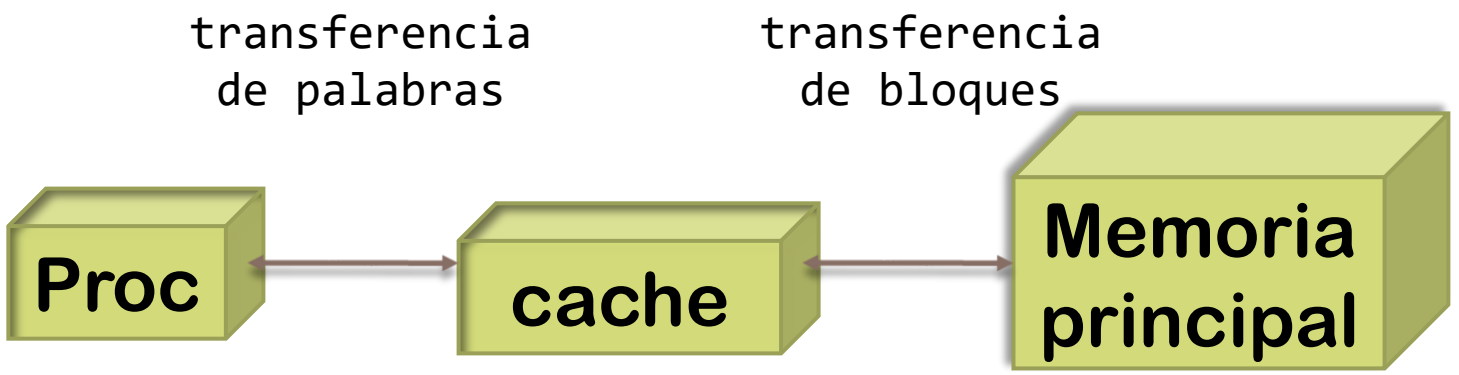
Objetivo de la memoria caché: aprovechar los accesos contiguos

- 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?

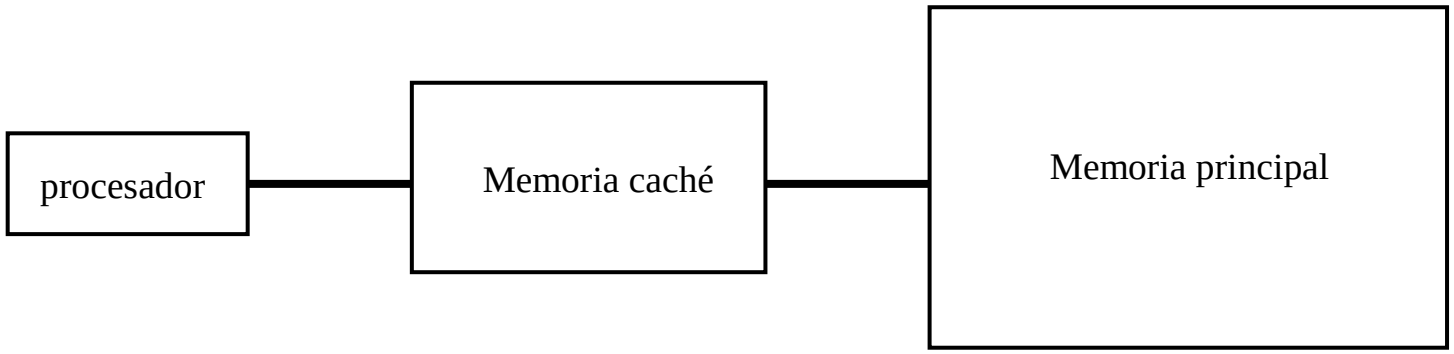


Memoria Cache

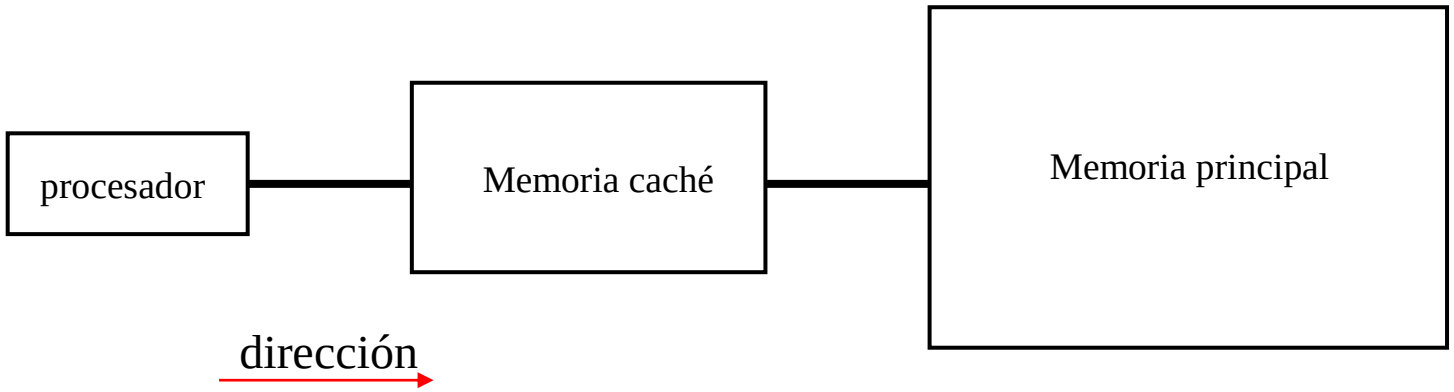
- Cantidad pequeña de memoria rápida SRAM
  - Integrada en el mismo procesador
  - Más rápida y cara que la memoria principal DRAM
- Está entre la memoria principal y el procesador
- Almacena una copia de partes de la memoria principal



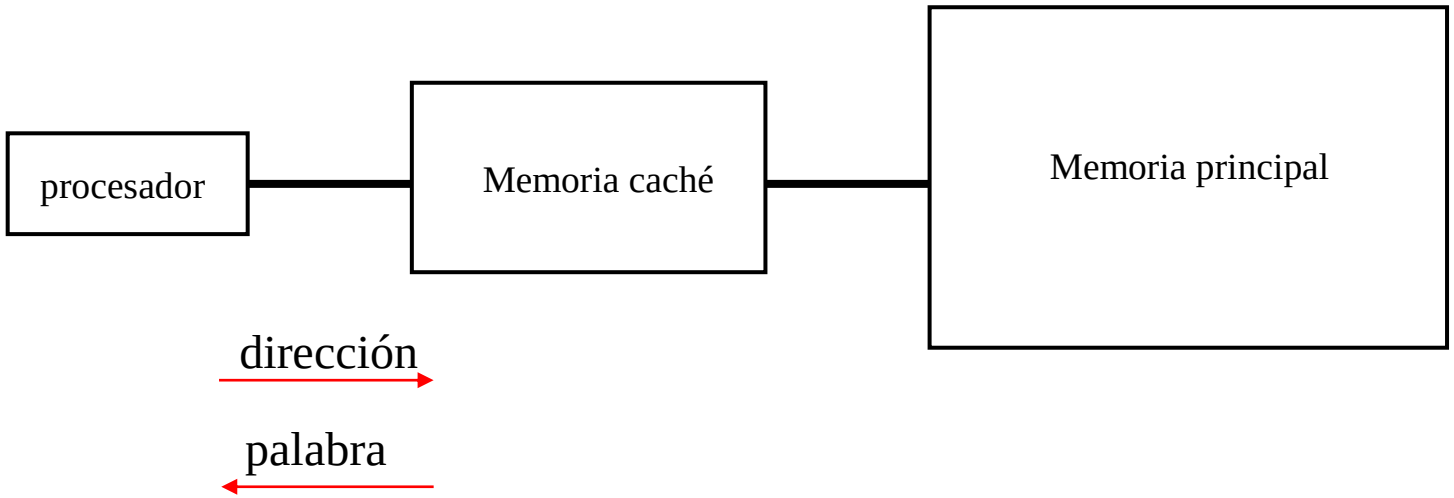
Funcionamieto de la memoria caché



Funcionamieto de la memoria caché

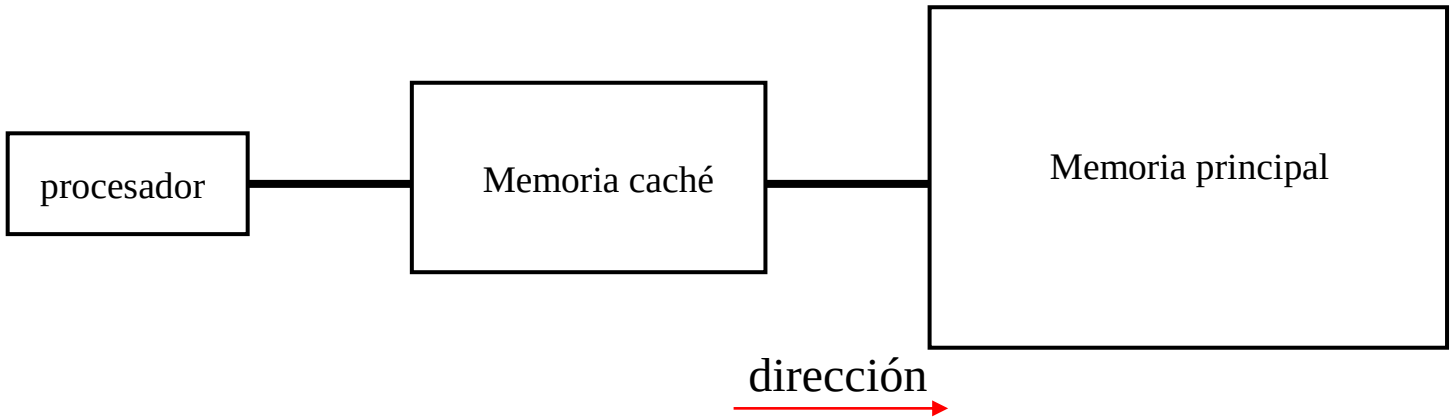


Funcionamieto de la memoria caché



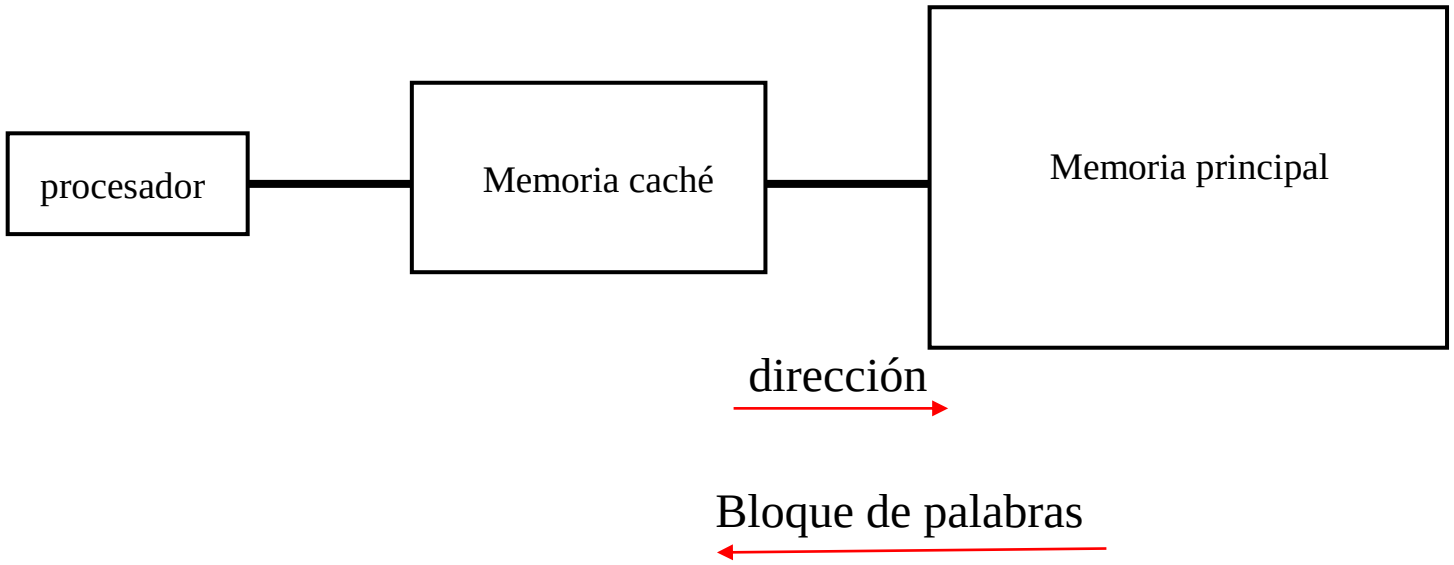
ACIERTO

Funcionamieto de la memoria caché

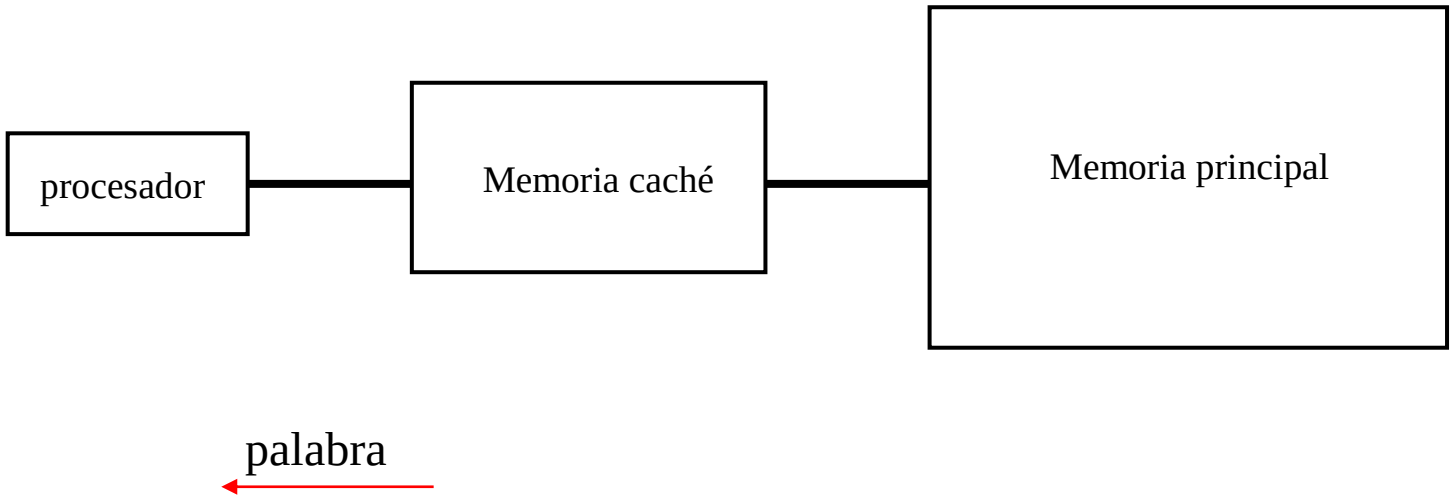


FALLO

Funcionamieto de la memoria caché

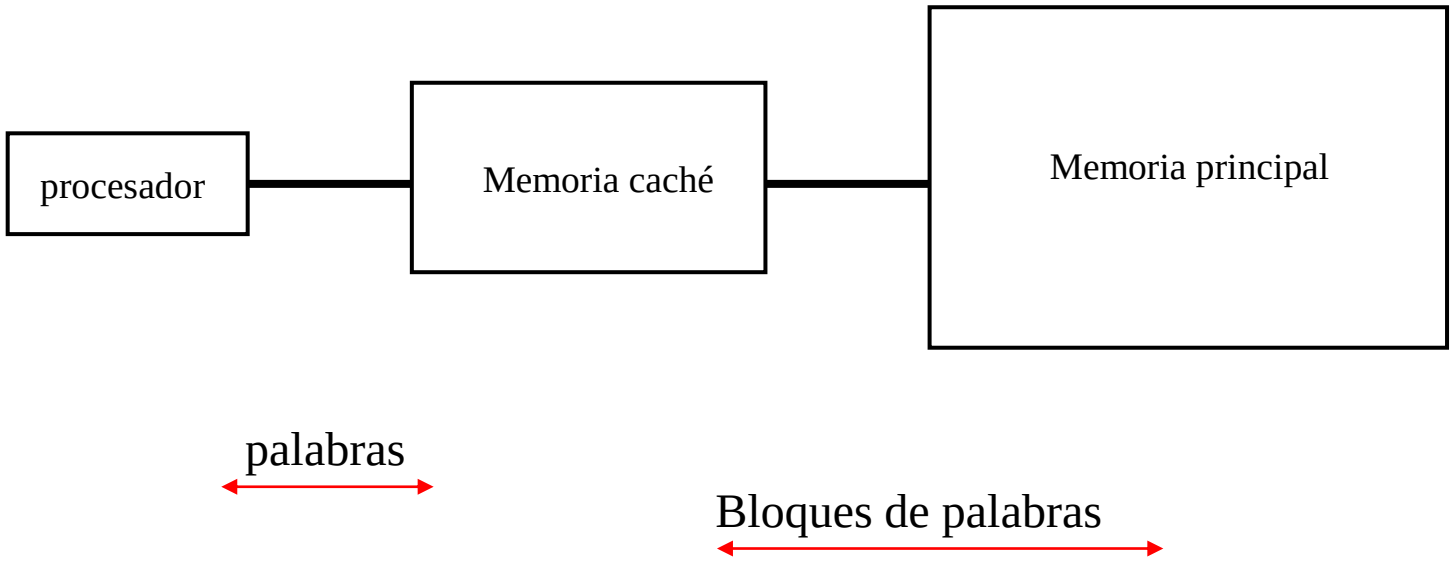


Funcionamieto de la memoria caché

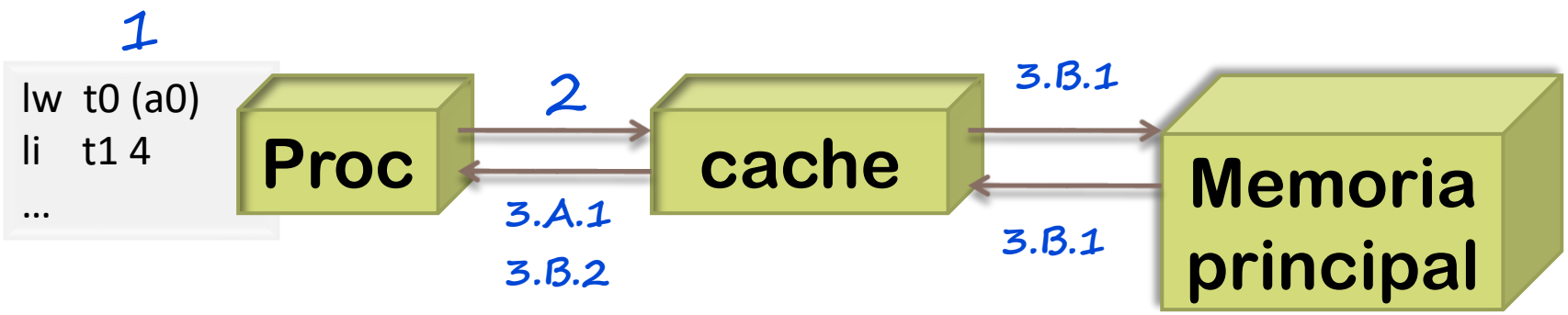


Funcionamieto de la memoria caché

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 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
          j    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
            j    bucle
fin:        …
```

- Sin memoria caché:
  - Número de accesos a memoria = 3 + 4 × 1000 + 1 = 4004 accesos
  - Tiempo de acceso a memoria = 4004 × 120 = 480480 ns = 480480 ms

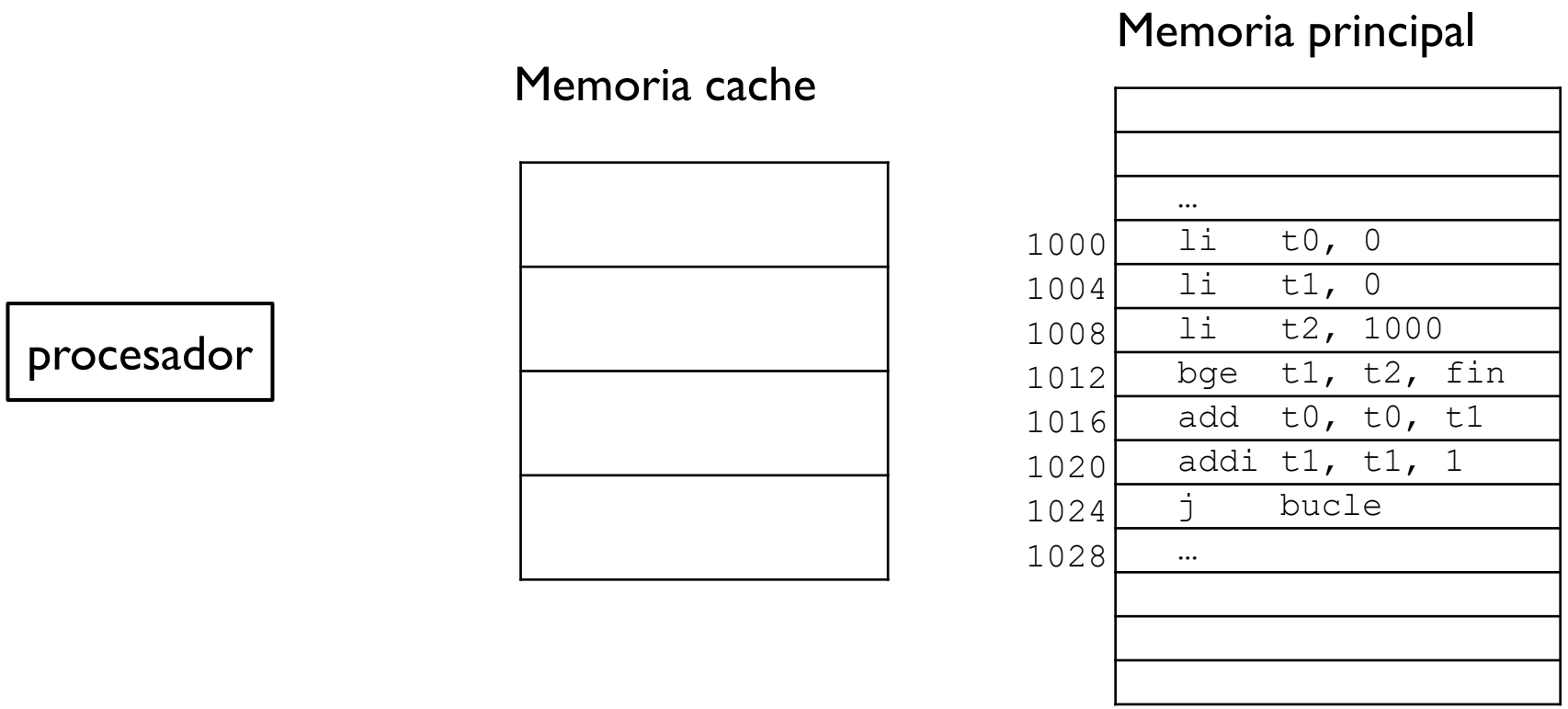
Ejemplo de funcionamiento

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

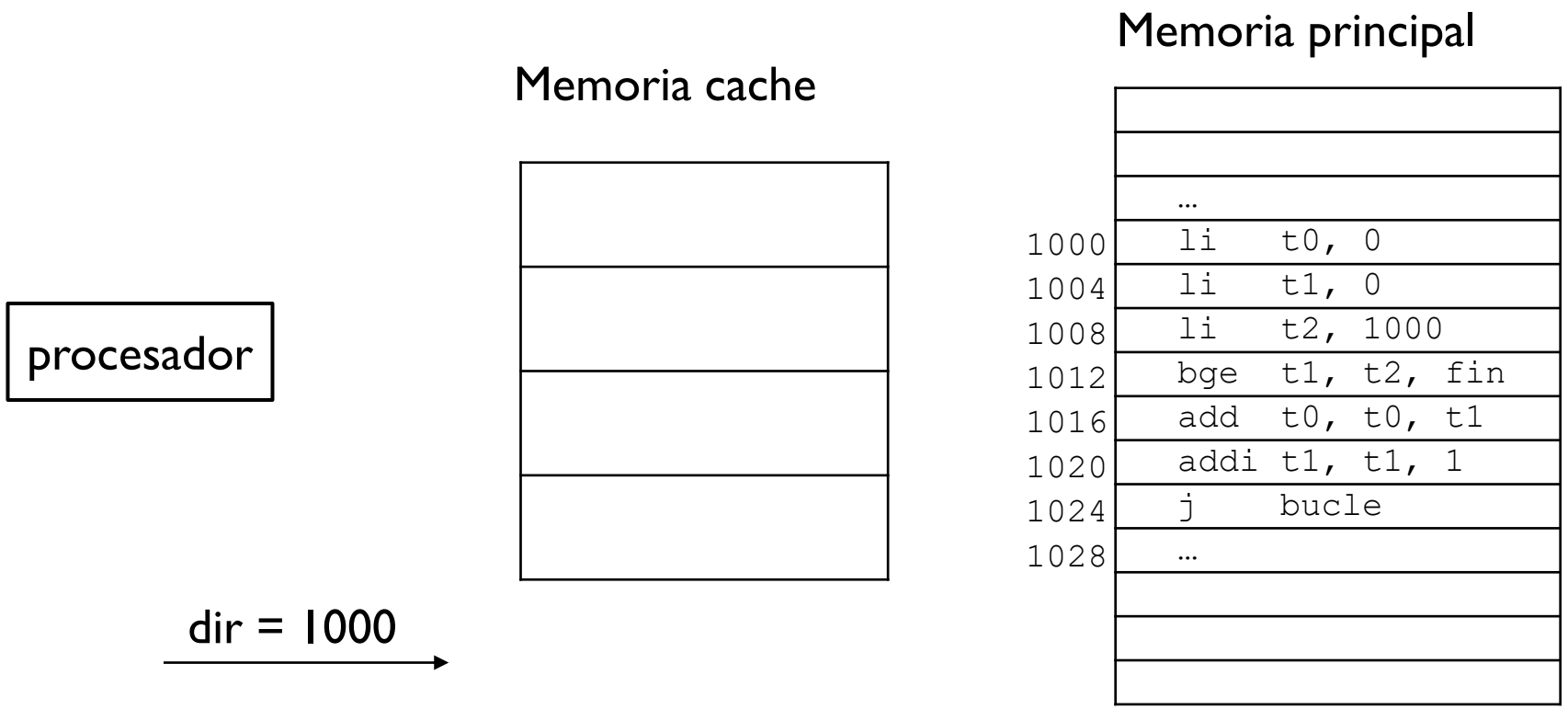
```
          li   t0, 0     # s
          li   t1, 0     # i
          li   t2, 1000
bucle:    bge  t1, t2, fin
          add  t0, t0, t1
          addi t1, t1, 1
          j    bucle
fin:      …
```

- Con memoria caché (bloque de 4 palabras):

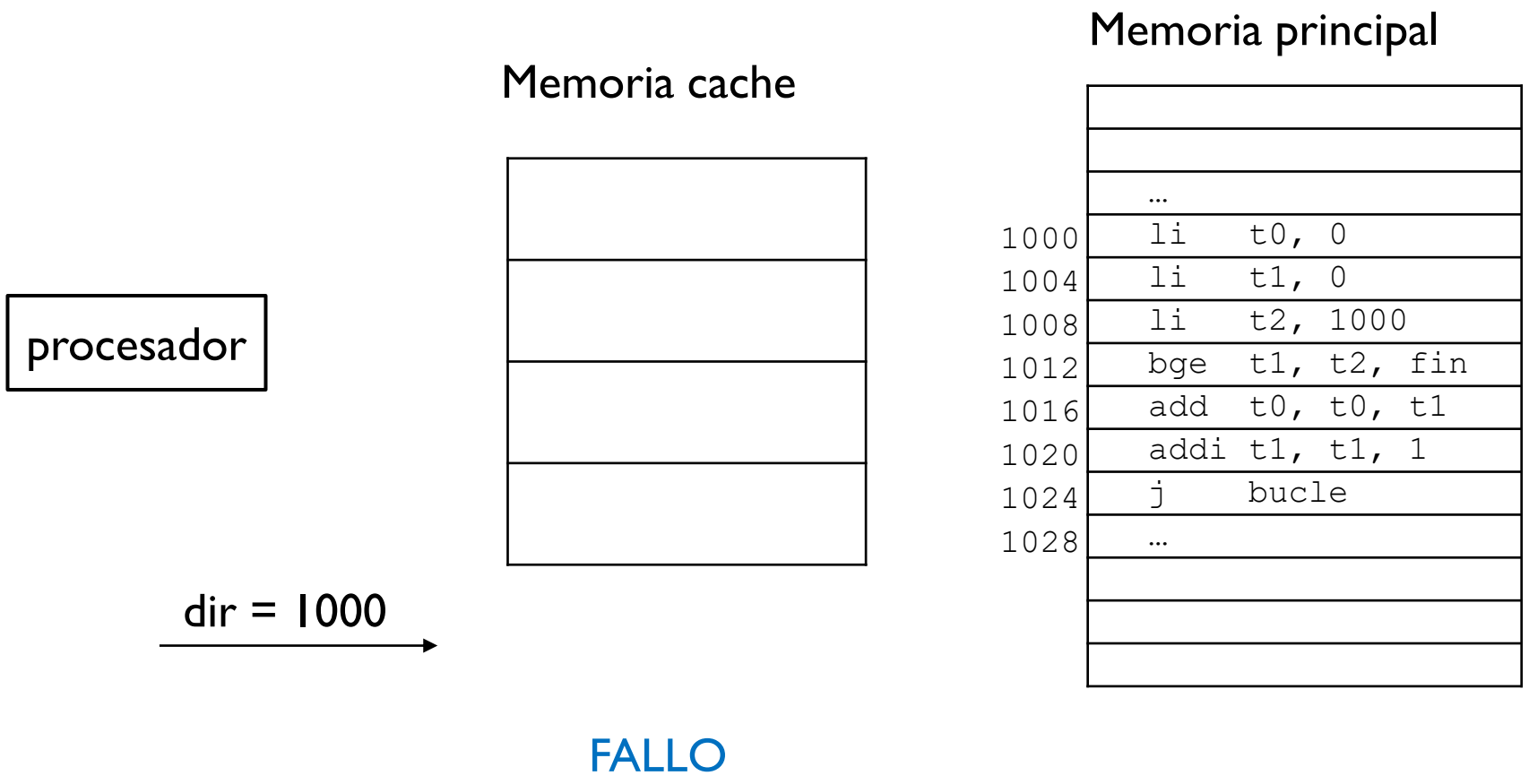
Ejemplo de funcionamiento



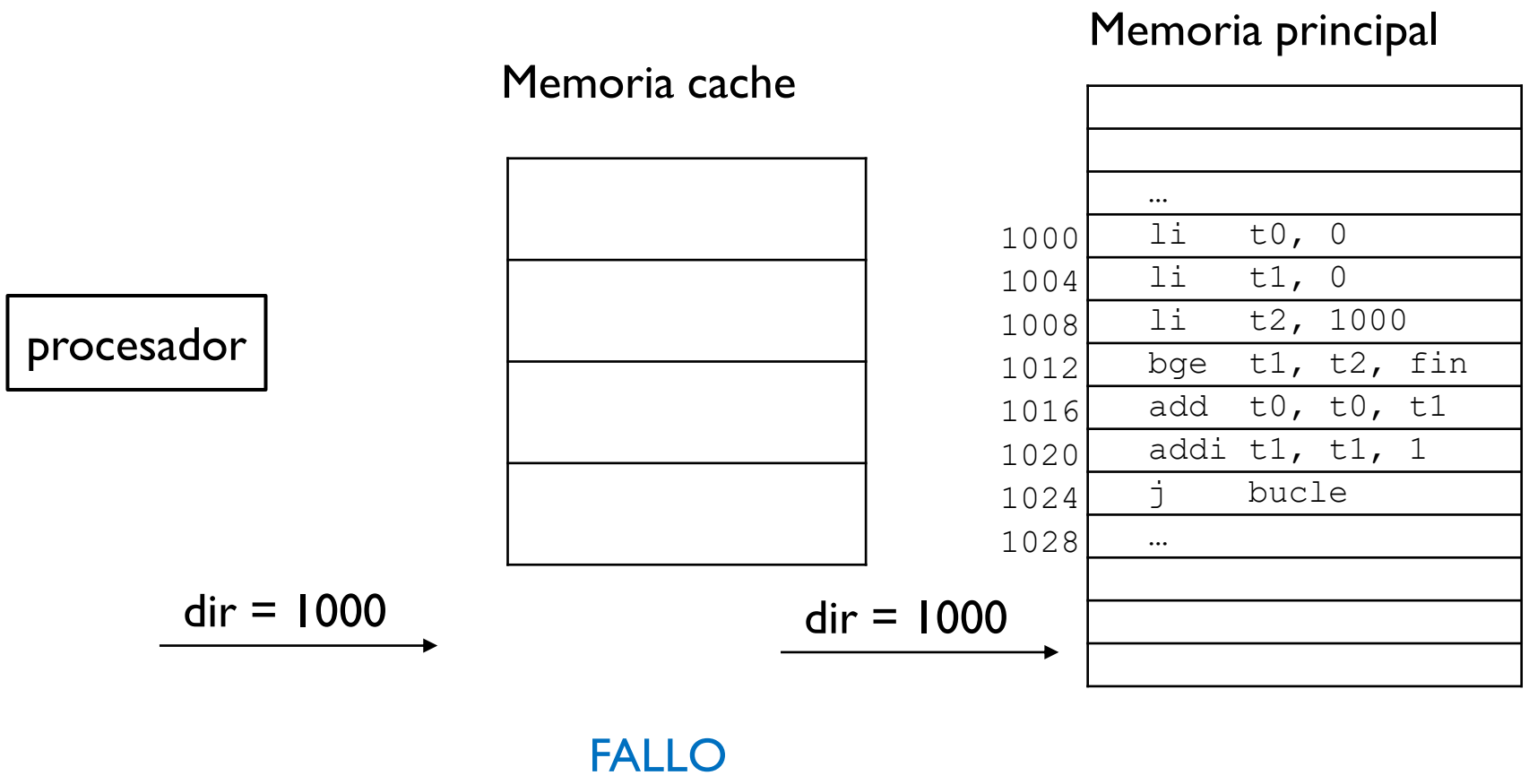
Ejemplo de funcionamiento



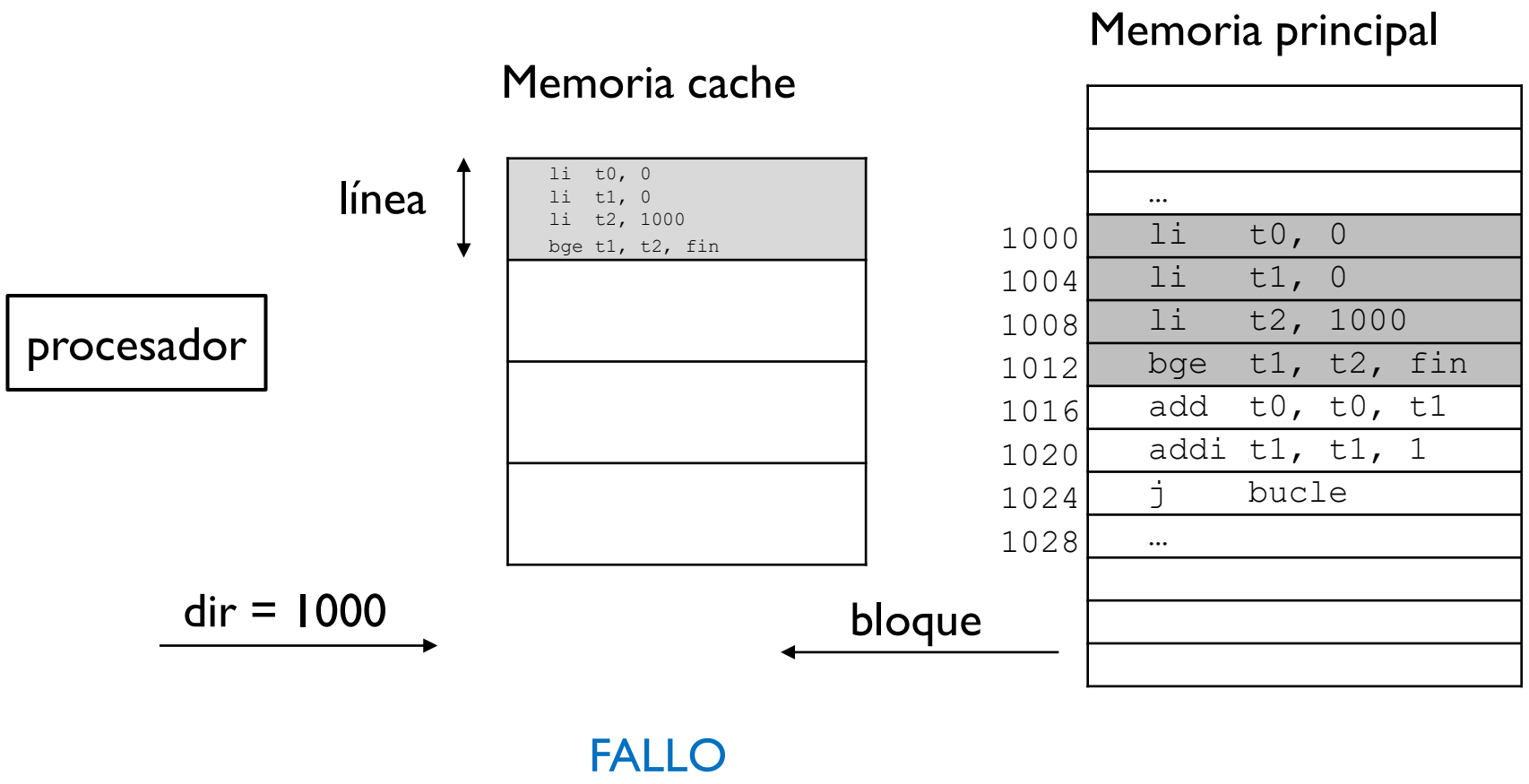
Ejemplo de funcionamiento



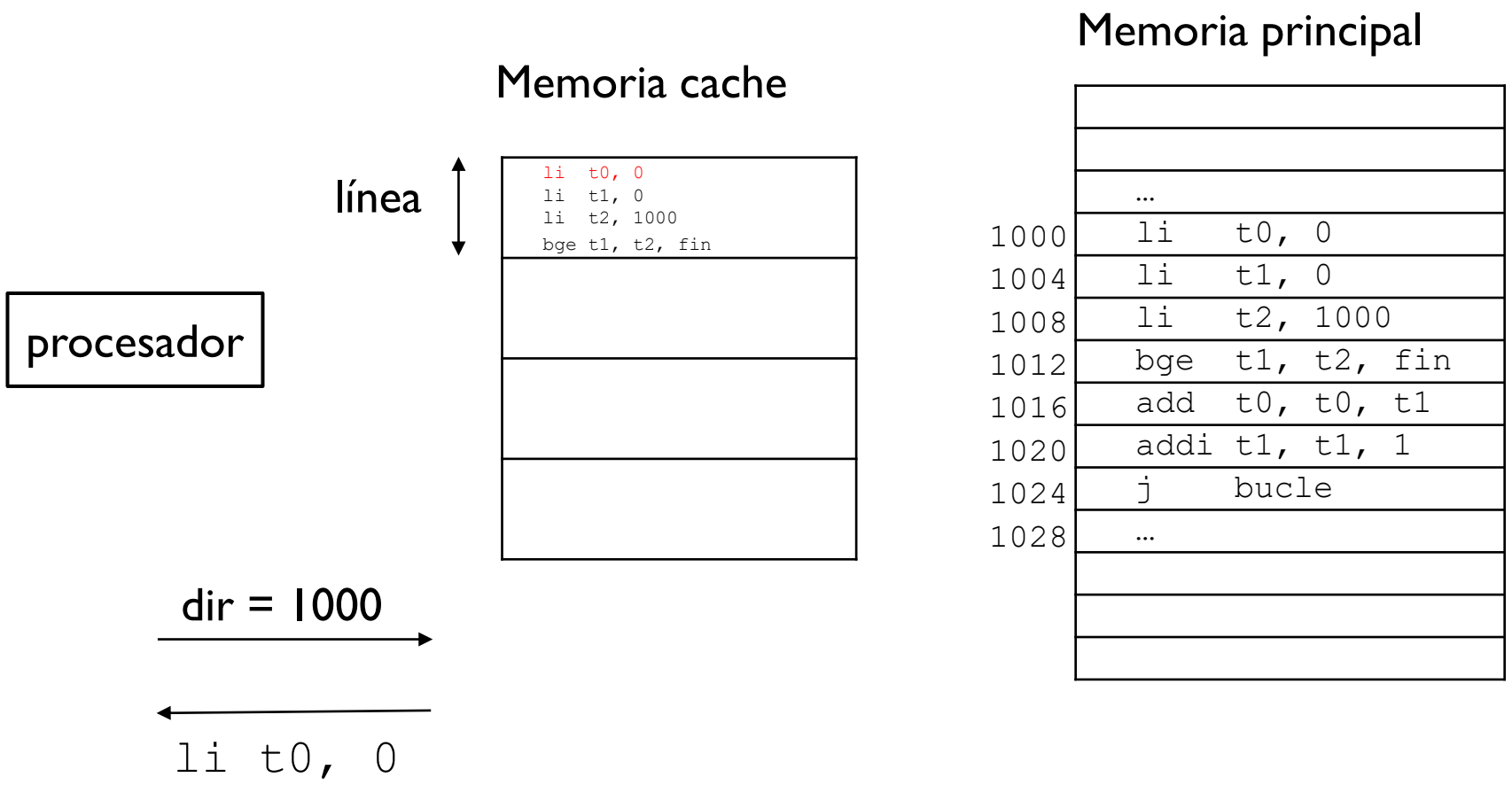
Ejemplo de funcionamiento



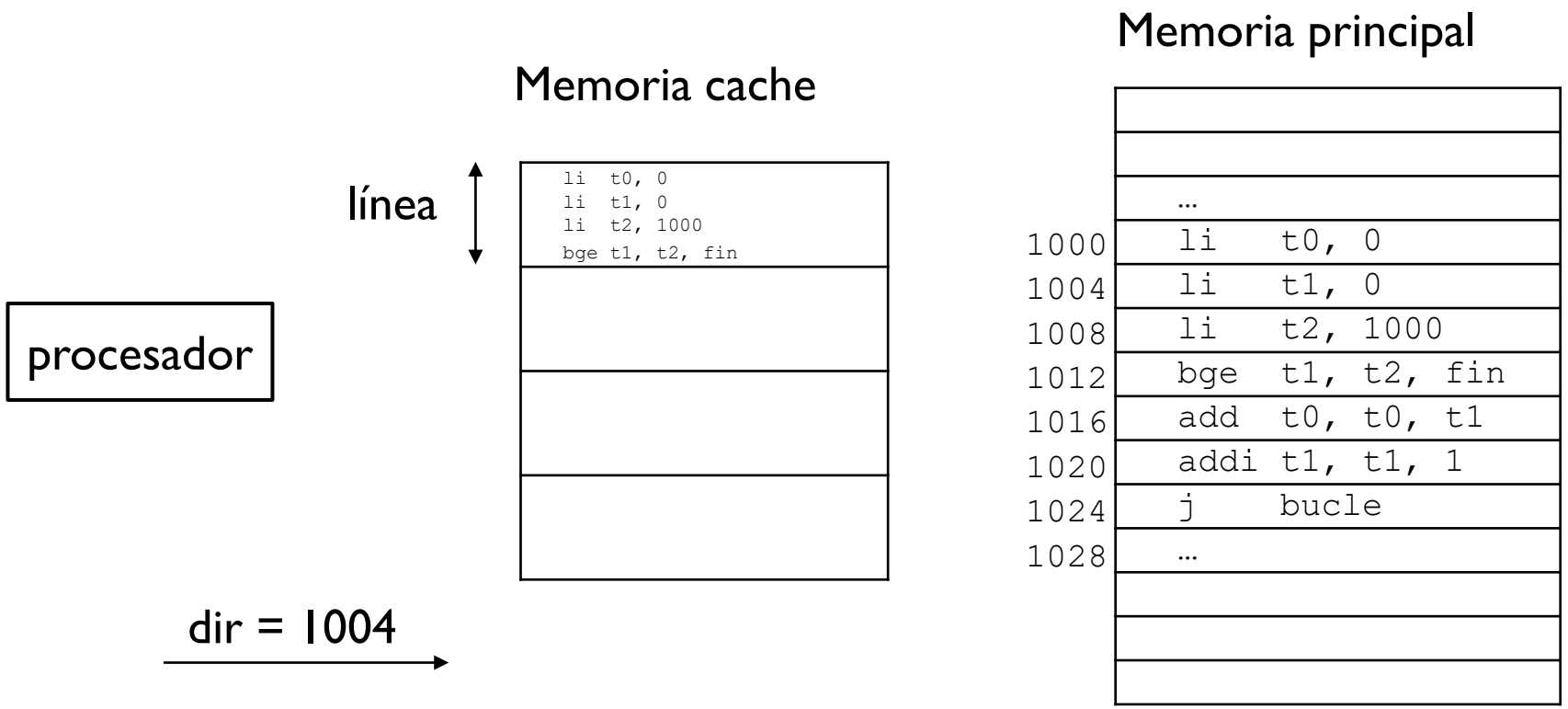
Ejemplo de funcionamiento



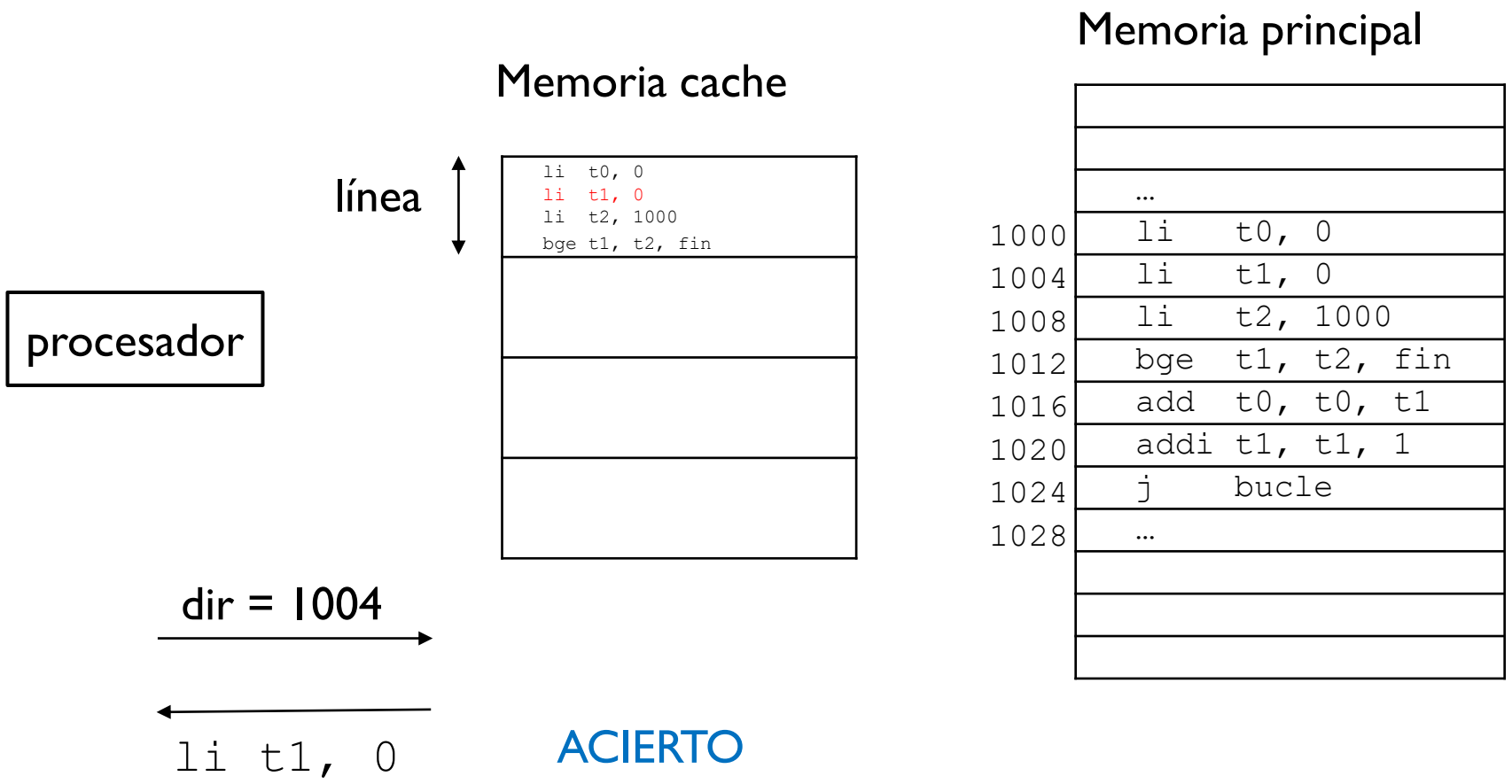
Ejemplo de funcionamiento



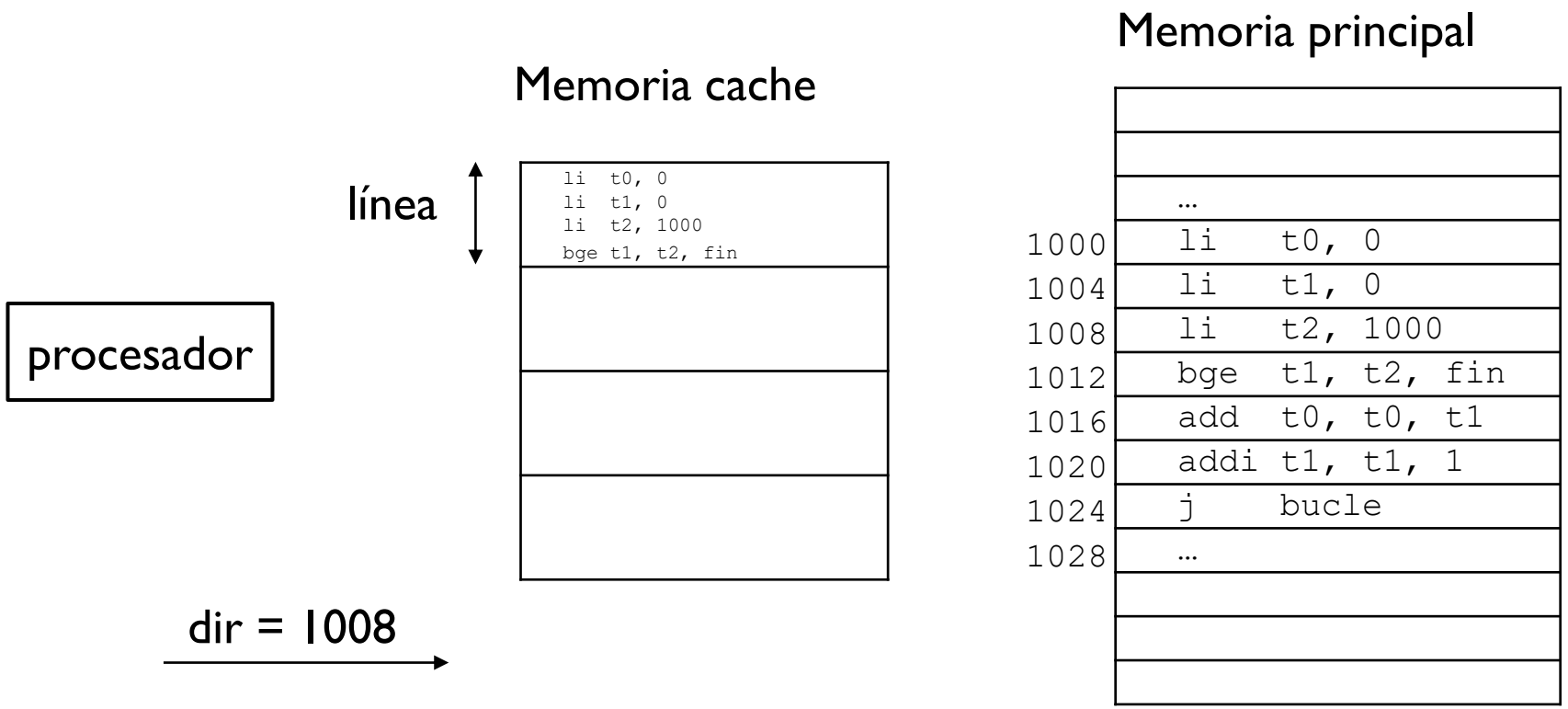
Ejemplo de funcionamiento



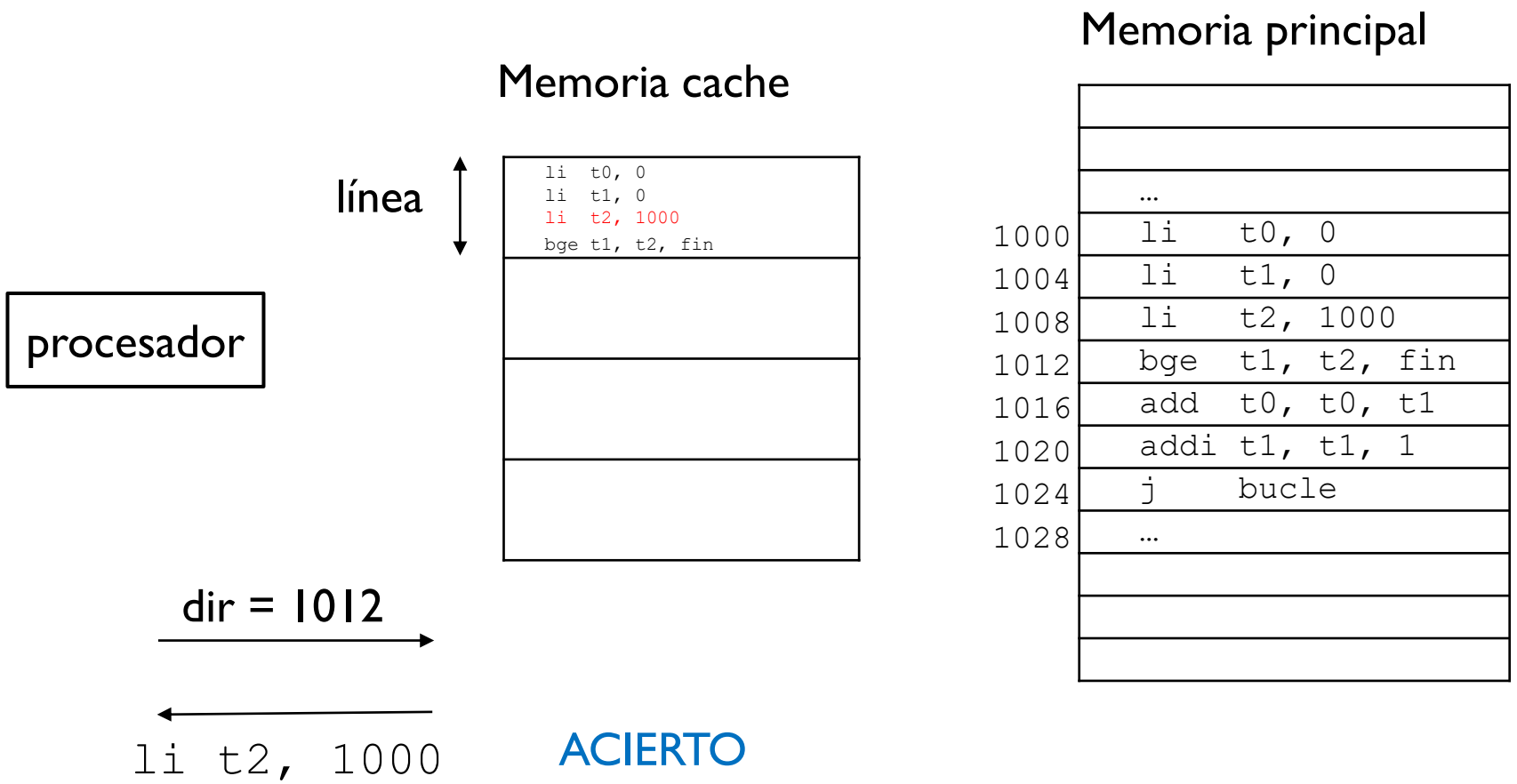
Ejemplo de funcionamiento



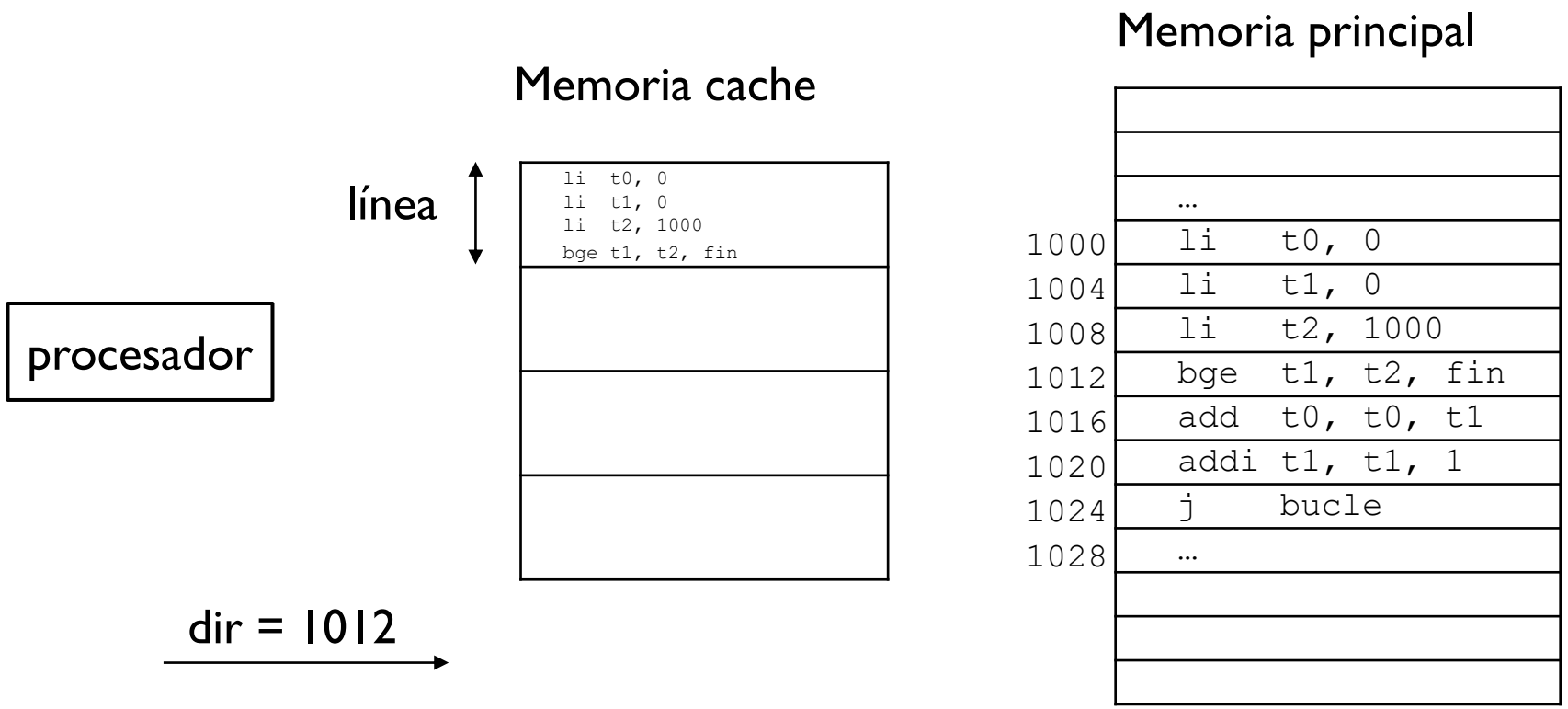
Ejemplo de funcionamiento



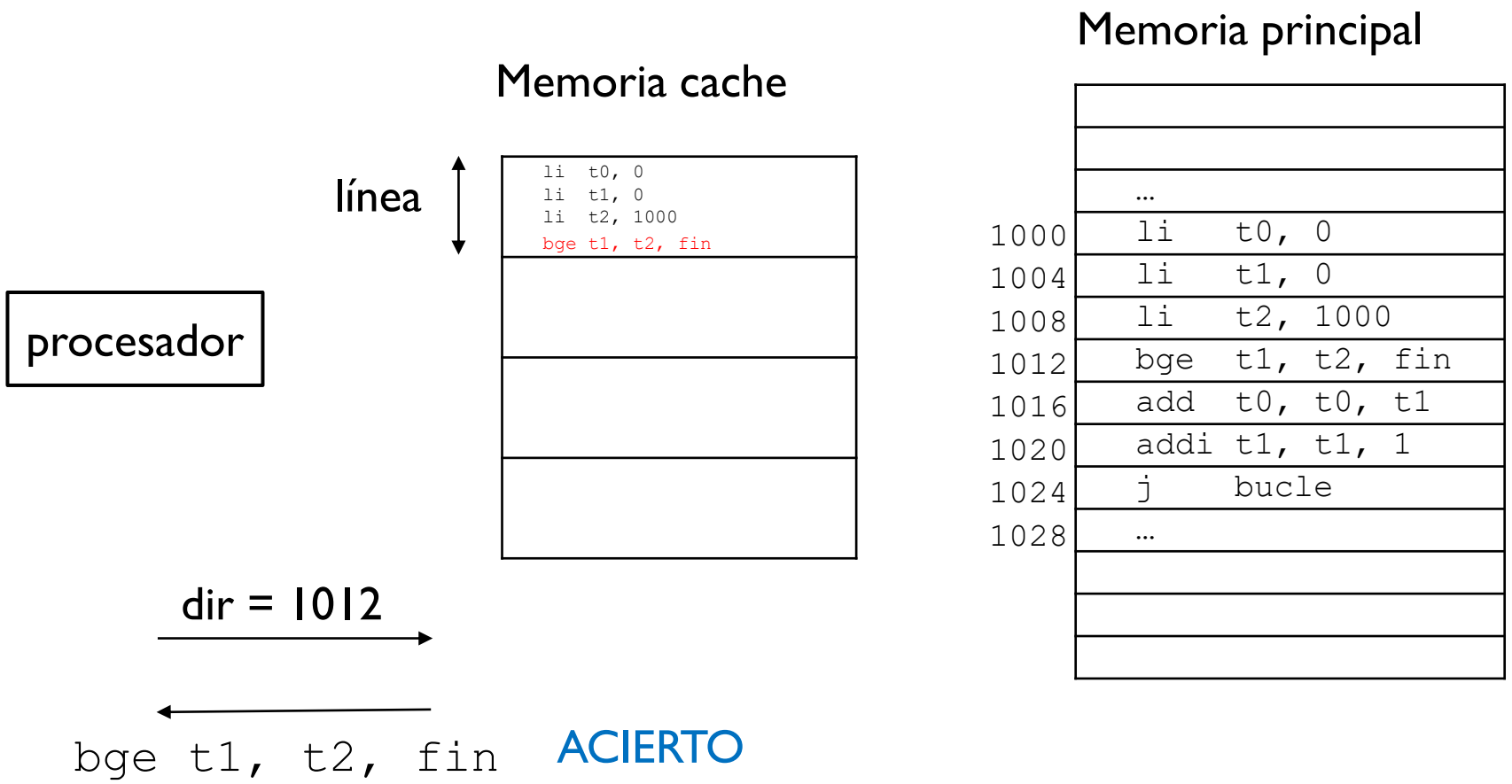
Ejemplo de funcionamiento



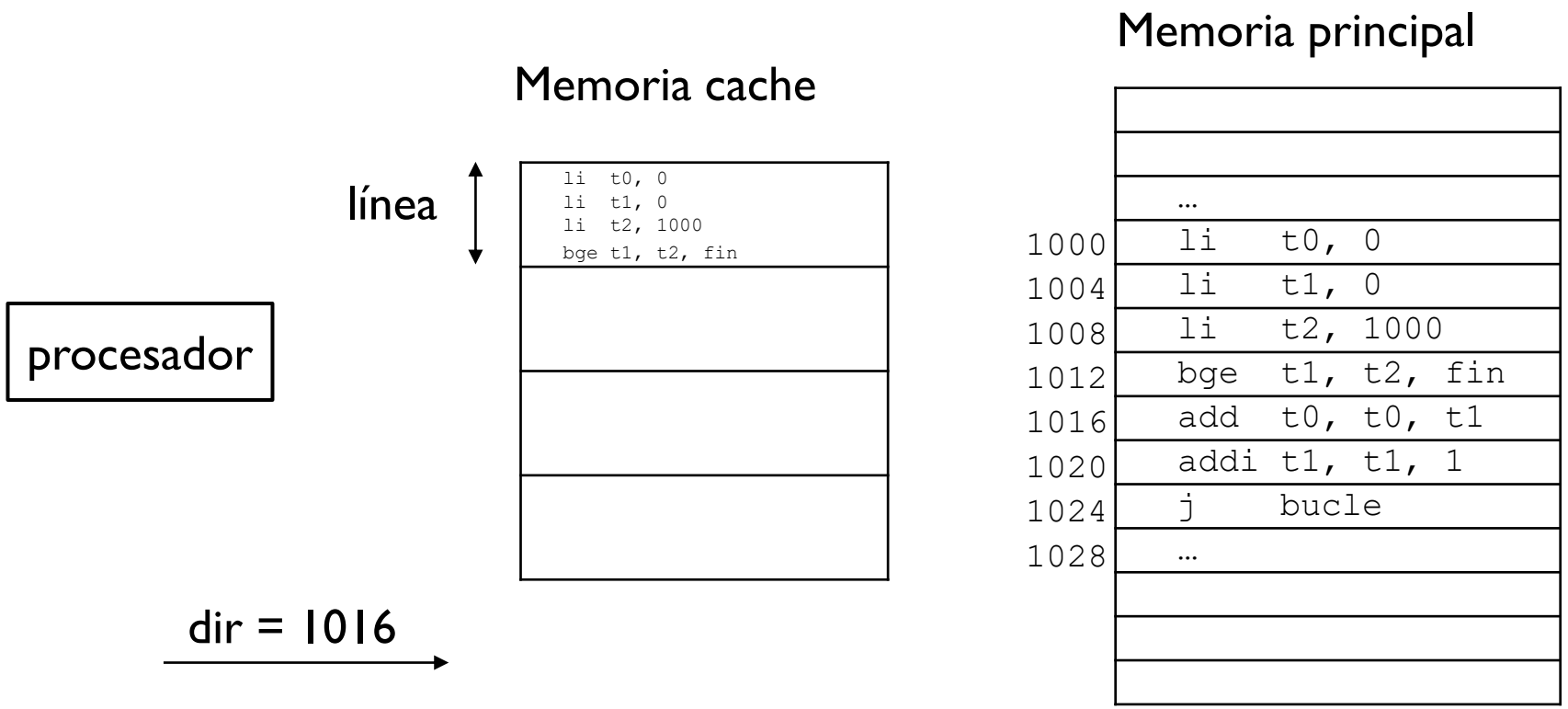
Ejemplo de funcionamiento



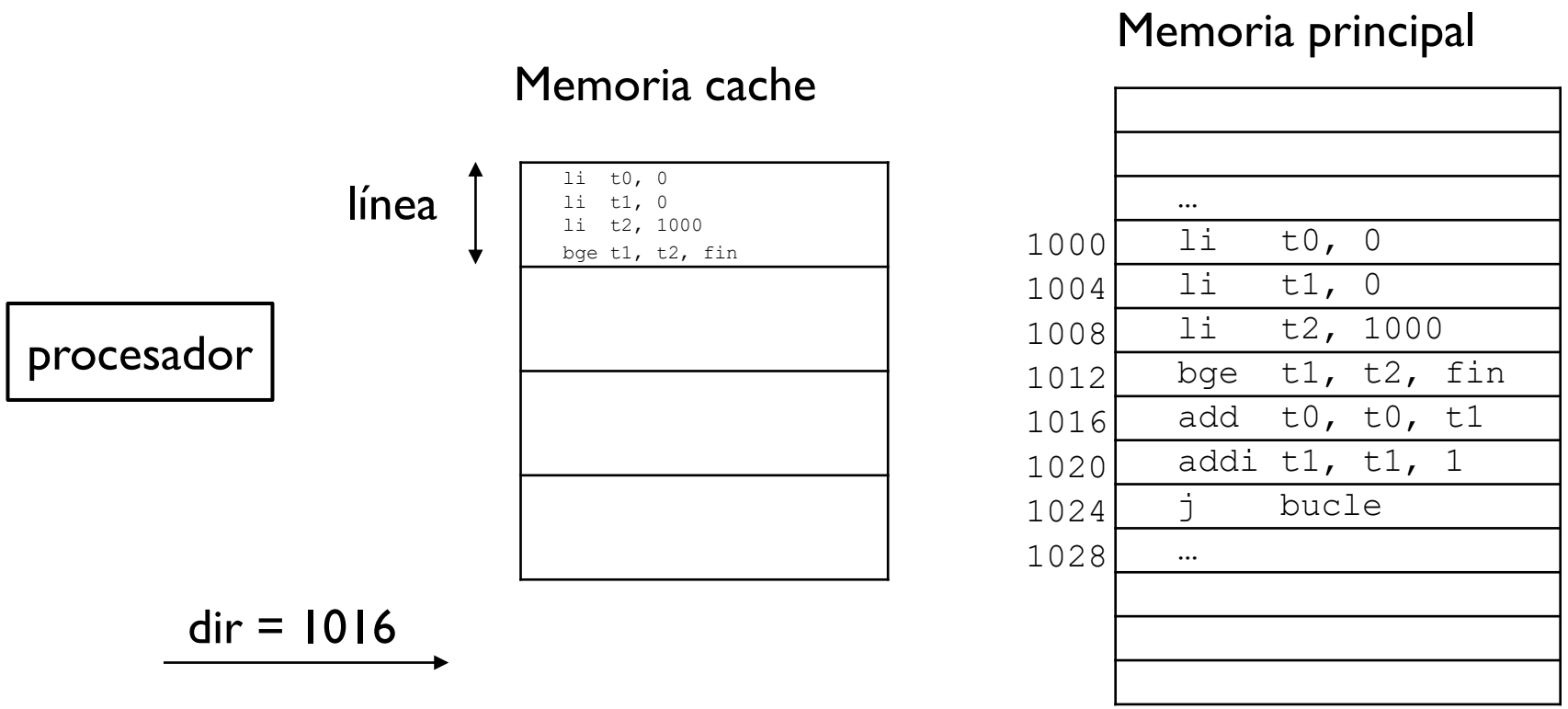
Ejemplo de funcionamiento



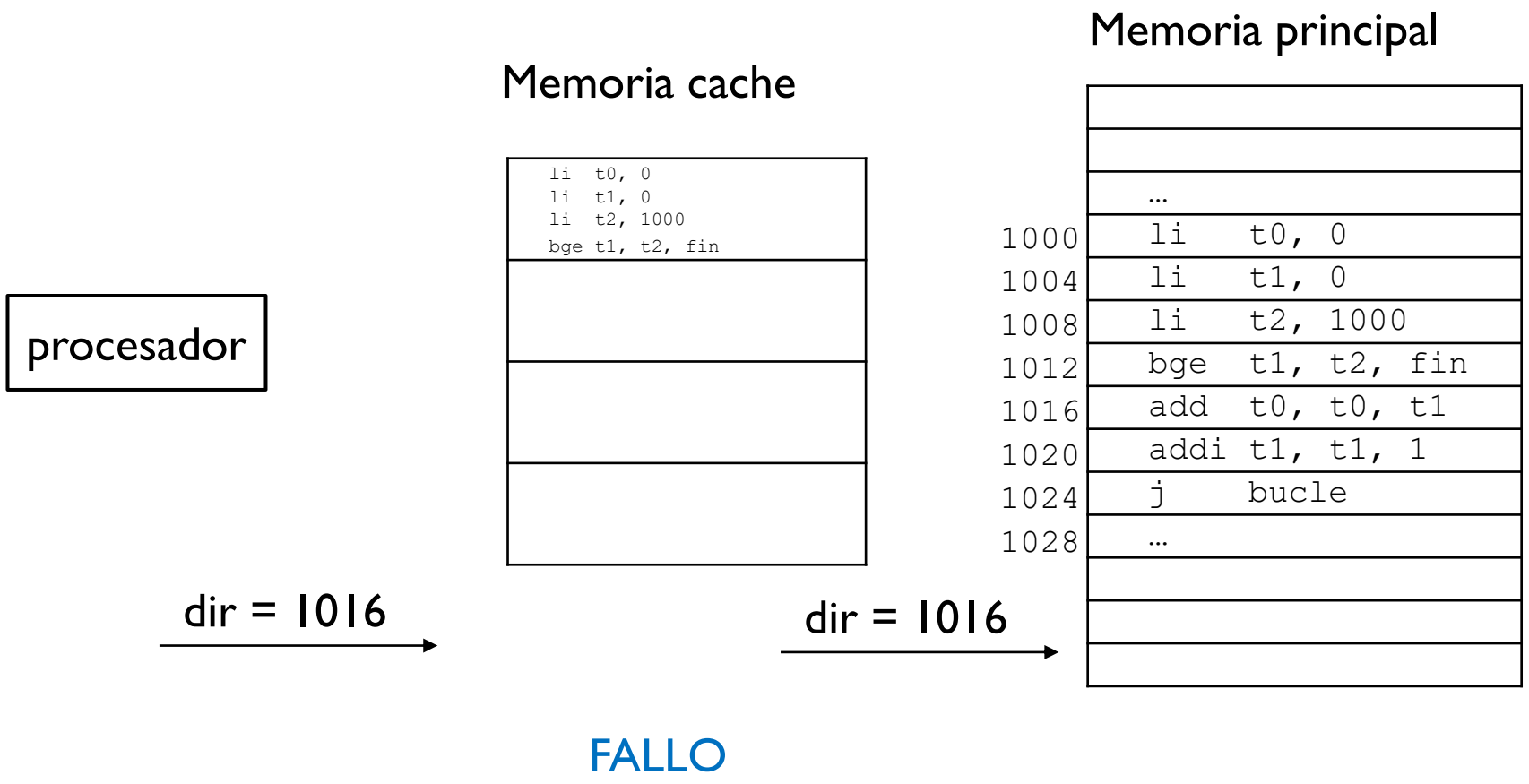
Ejemplo de funcionamiento



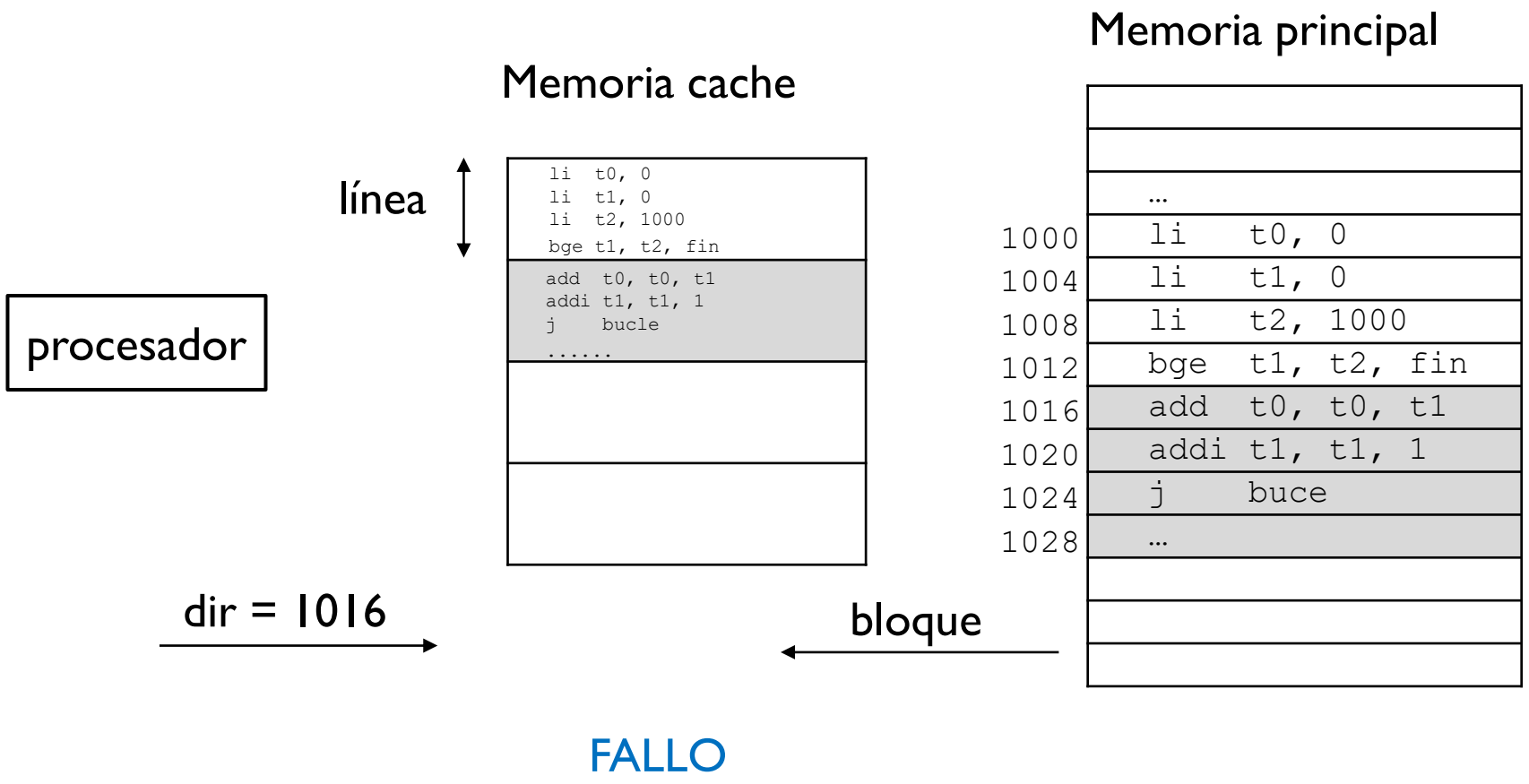
Ejemplo de funcionamiento



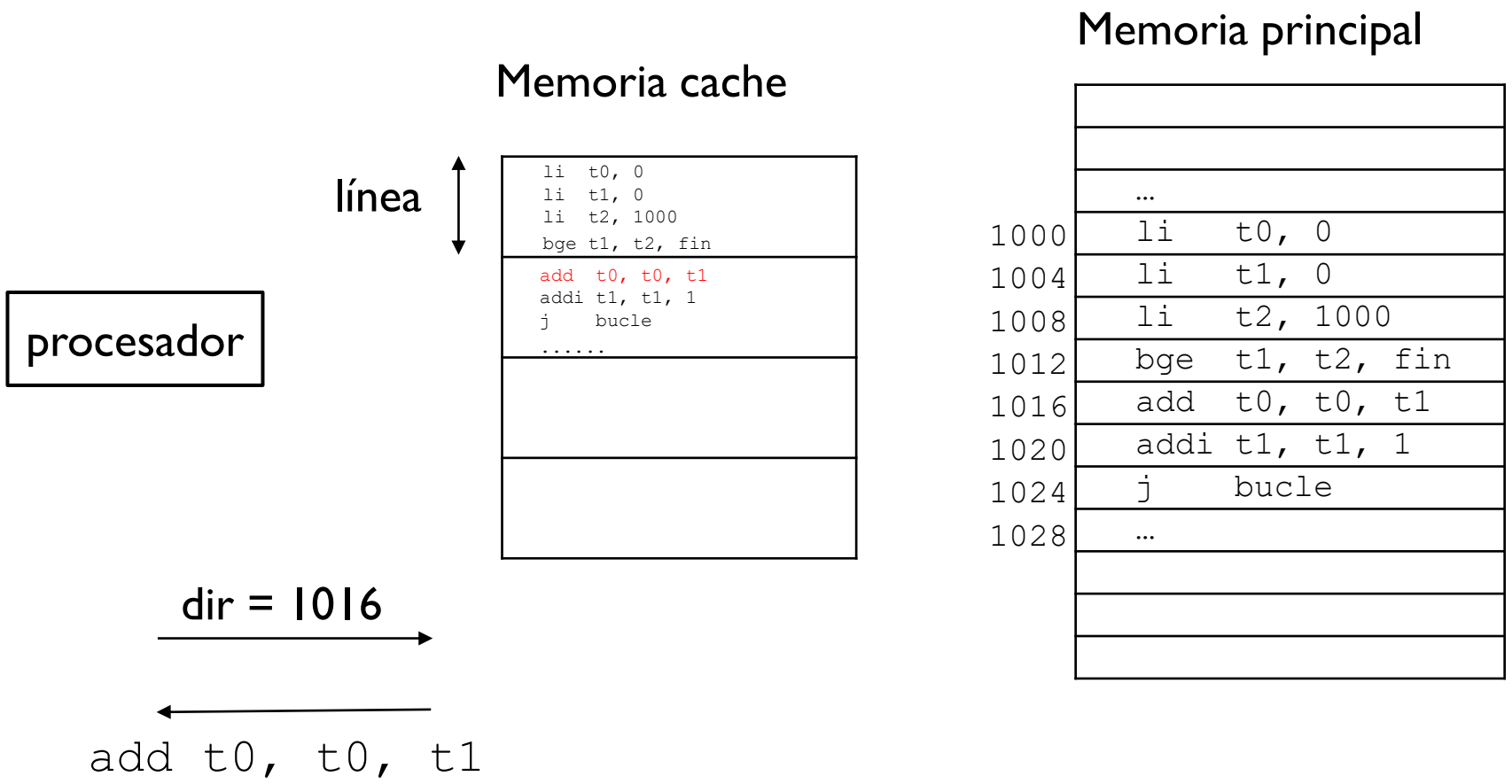
Ejemplo de funcionamiento



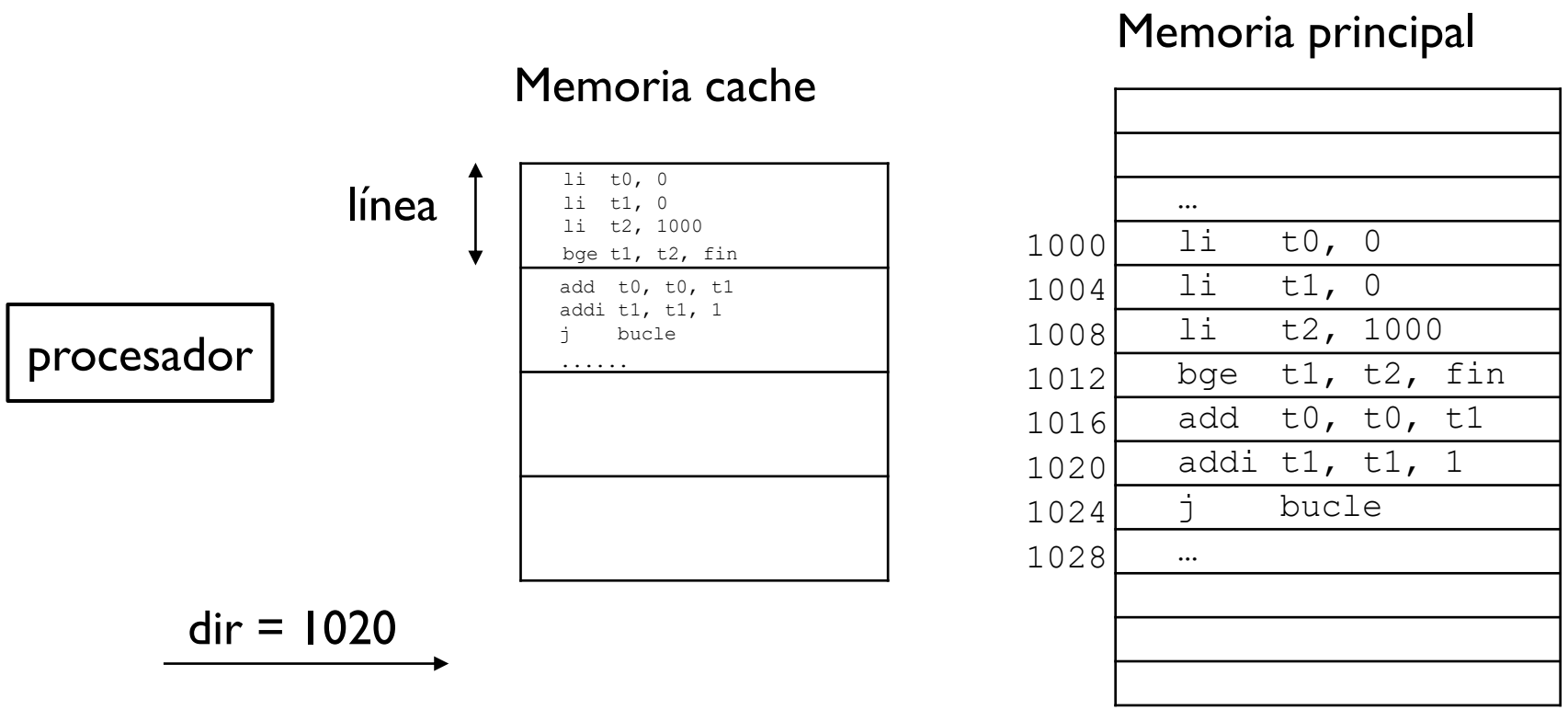
Ejemplo de funcionamiento



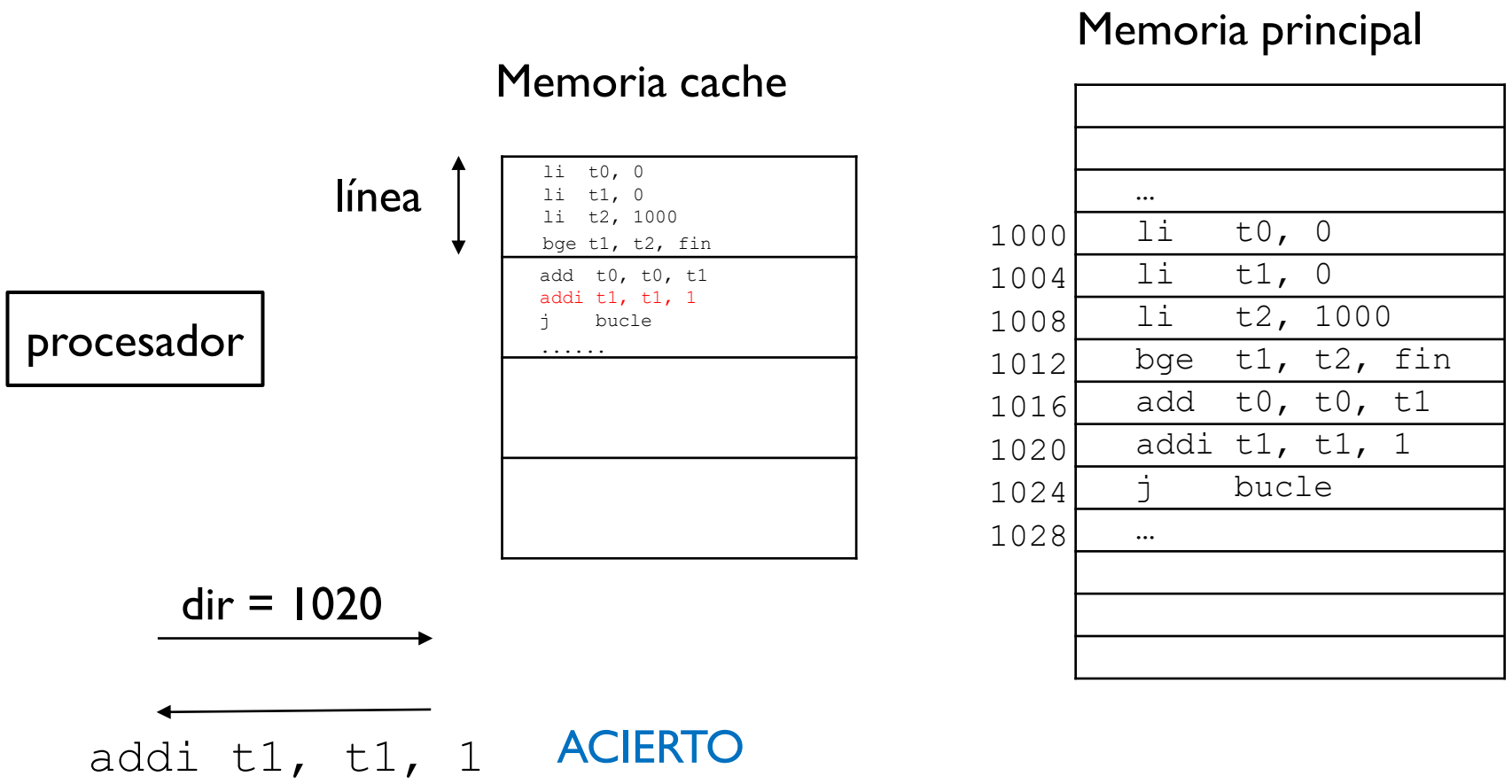
Ejemplo de funcionamiento



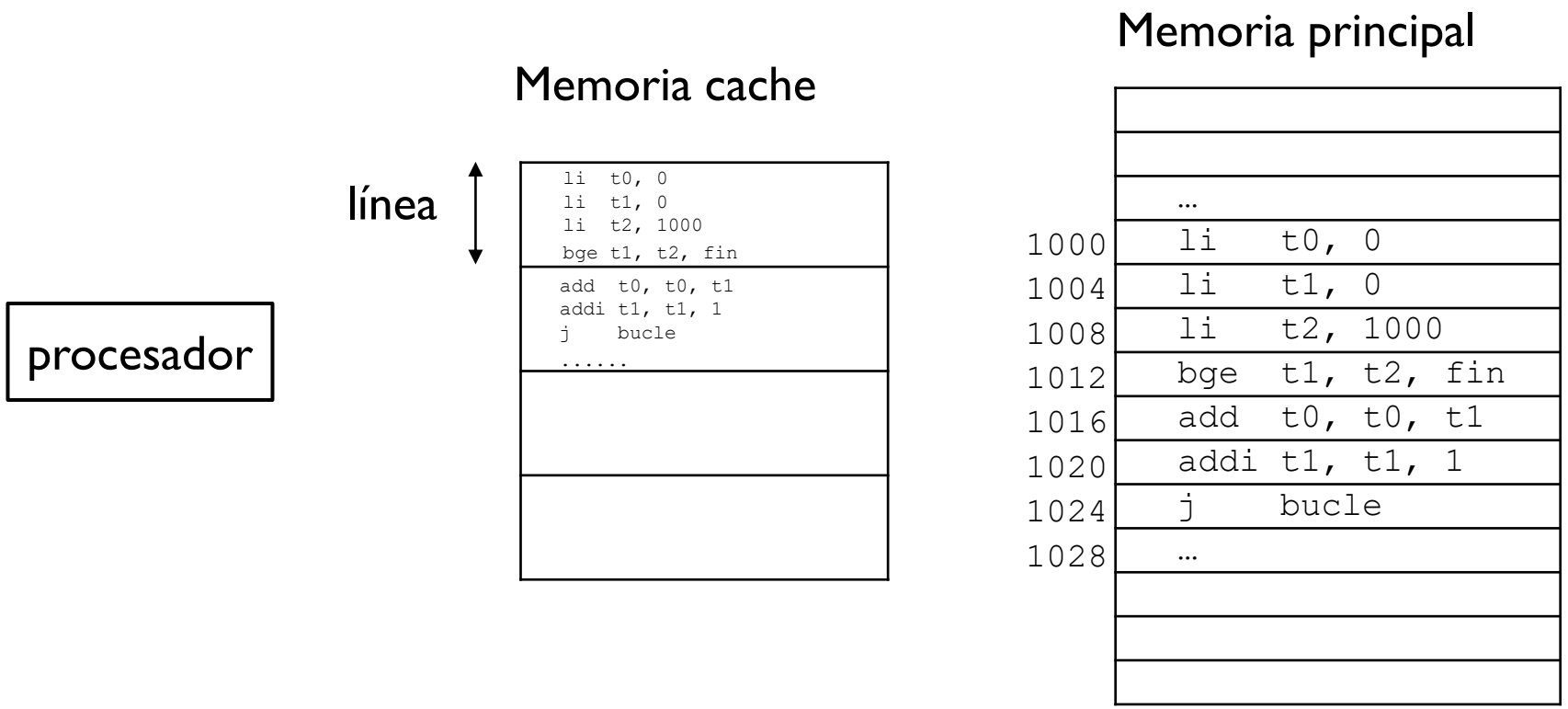
Ejemplo de funcionamiento



Ejemplo de funcionamiento



Ejemplo de funcionamiento



El resto de los accesos son ACIERTOS

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
        j    bucle
fin:    …
```

- Con memoria caché (bloque de 4 palabras):
  - Número de bloques = 2
  - Número de fallos = 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
        j    bucle
fin:    …
```

- Con memoria caché:
  - Número de bloques = 2
  - Número de fallos = 2
  - Tiempo para transferir 2 bloques = 200 × 2 = 400 ns
  - Tiempo de acceso a la caché = 4004 × 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
        j    bucle
fin:    …
```

- Sin memoria caché = 480480 ns
- Con memoria caché = 8408 ns
- Tasa de aciertos a la caché = 4002 / 4004 => 99,95 %

Ejercicio
Calcular tasa de aciertos

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

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

- 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

```
.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
       j    bucle
fin:   …
```

¿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

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

- Ta: tiempo de acceso a la caché
- Tf: tiempo en tratar el fallo, tiempo en remplazar un bloque y traerlo de memoria principal a caché
- h: tasa de aciertos

Rendimiento general

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

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): Ta
   - **Si no está (FALLO)**,
     - **3.B.1** La cache transfiere de memoria principal el bloque asociado a la posición: Tf
     - **3.B.2** Después, la cache entrega los datos pedidos al procesador : Ta

Ejemplo

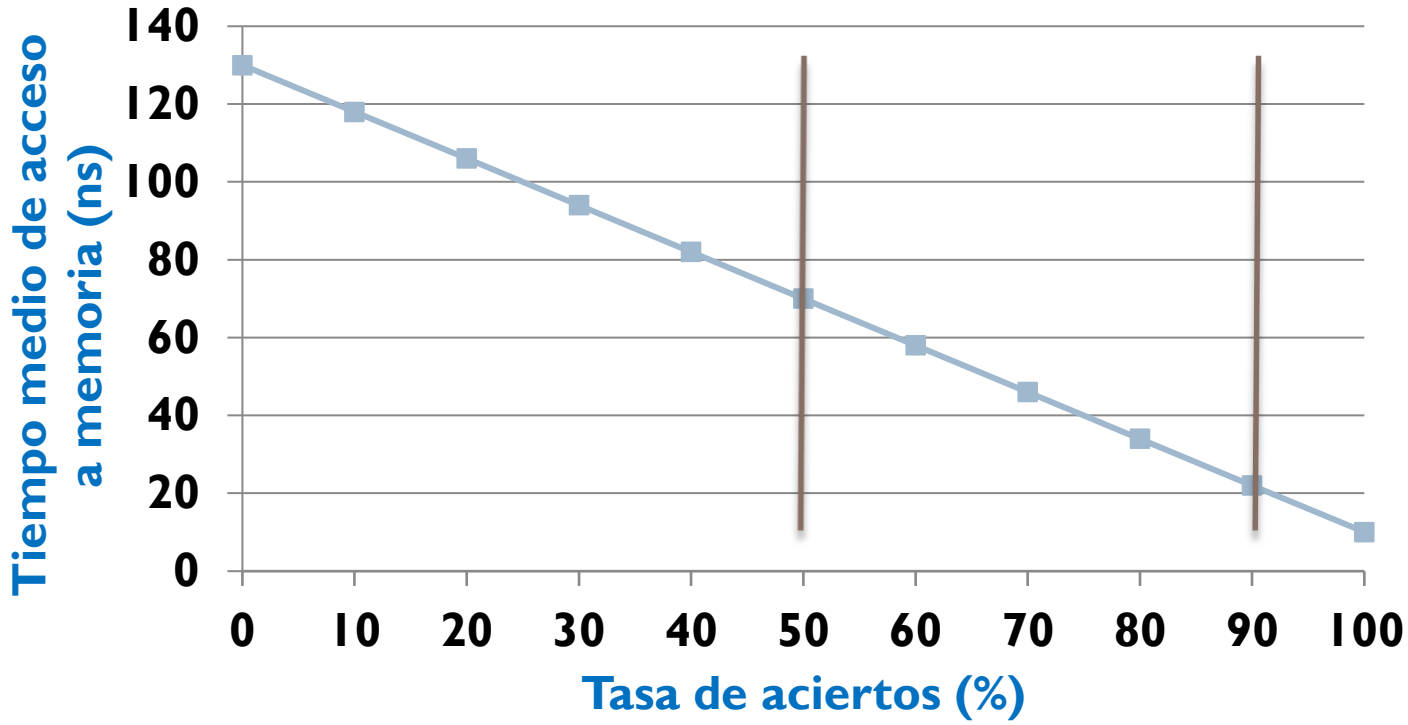
$$Tm = h \cdot Ta + (1-h) \cdot (Ta+Tf)$$
$$= Ta + (1-h) \cdot Tf$$

1. Ta: Tiempo de acceso a caché = **10** ns
2. Tf: Tiempo de acceso a memoria principal = **120** ns
3. h: tasa de aciertos-> **X = 0.1, 0.2, …, 0.9, 1.0**

10%, 20%, …, 90%, 100%

Ejemplo

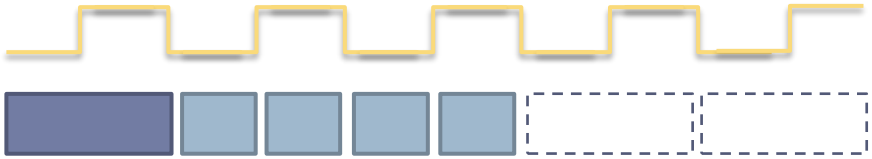
$$Tm = h \cdot Ta + (1-h) \cdot (Ta+Tf)$$
$$= Ta + (1-h) \cdot Tf$$



Acceso por bloques

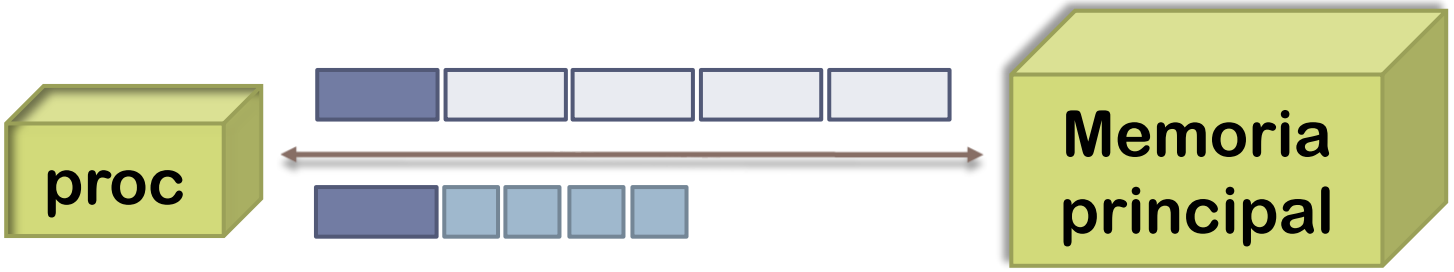
- Se premia el acceso a posiciones consecutivas de memoria
  - Ejemplo 1:
    acceder a 5 posiciones de memoria individuales (no consecutivas)

  - 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 separarase 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 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?

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

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

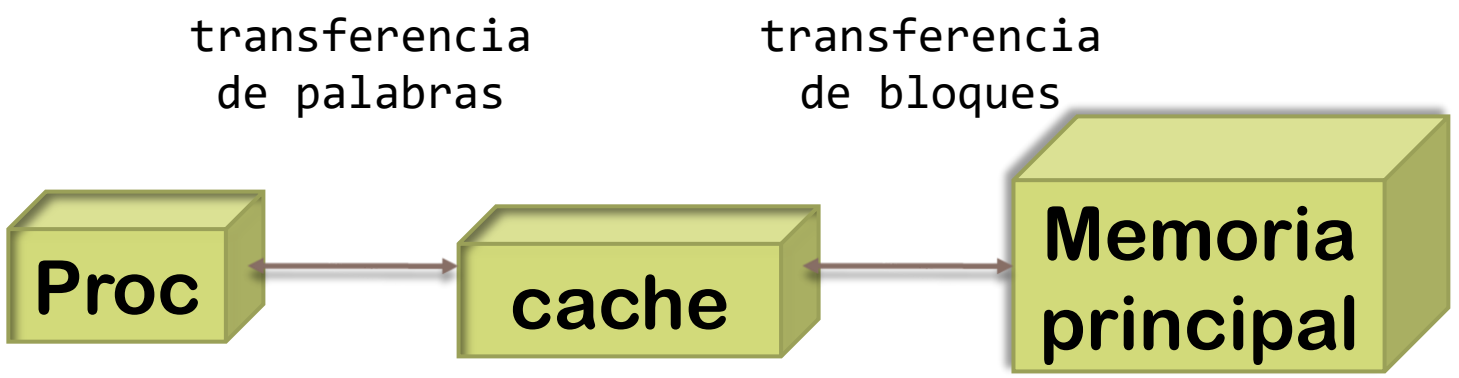
$$5 = 4 \times h + (120 + 4) \times (1 - h) \qquad \Rightarrow$$

$$\Rightarrow h > 0.9916$$

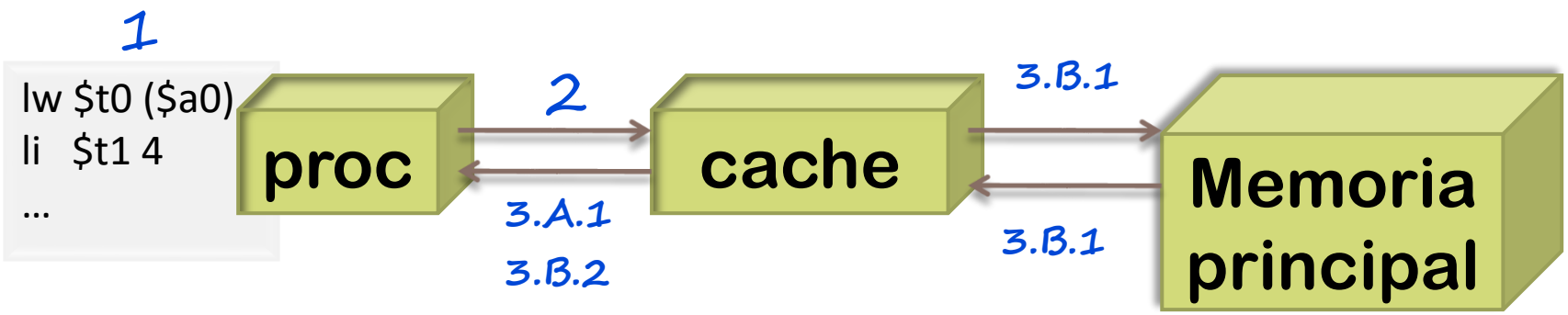
Memoria caché
Resumen

- 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.

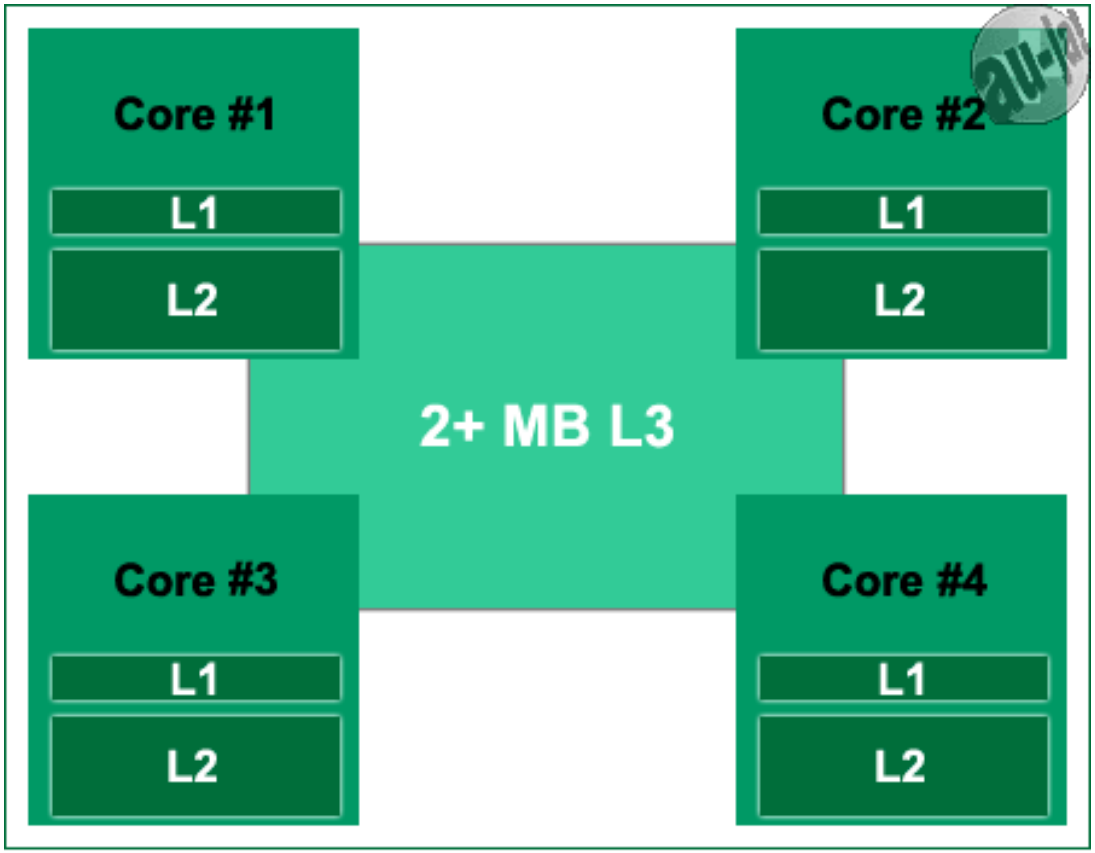


Funcionamiento general



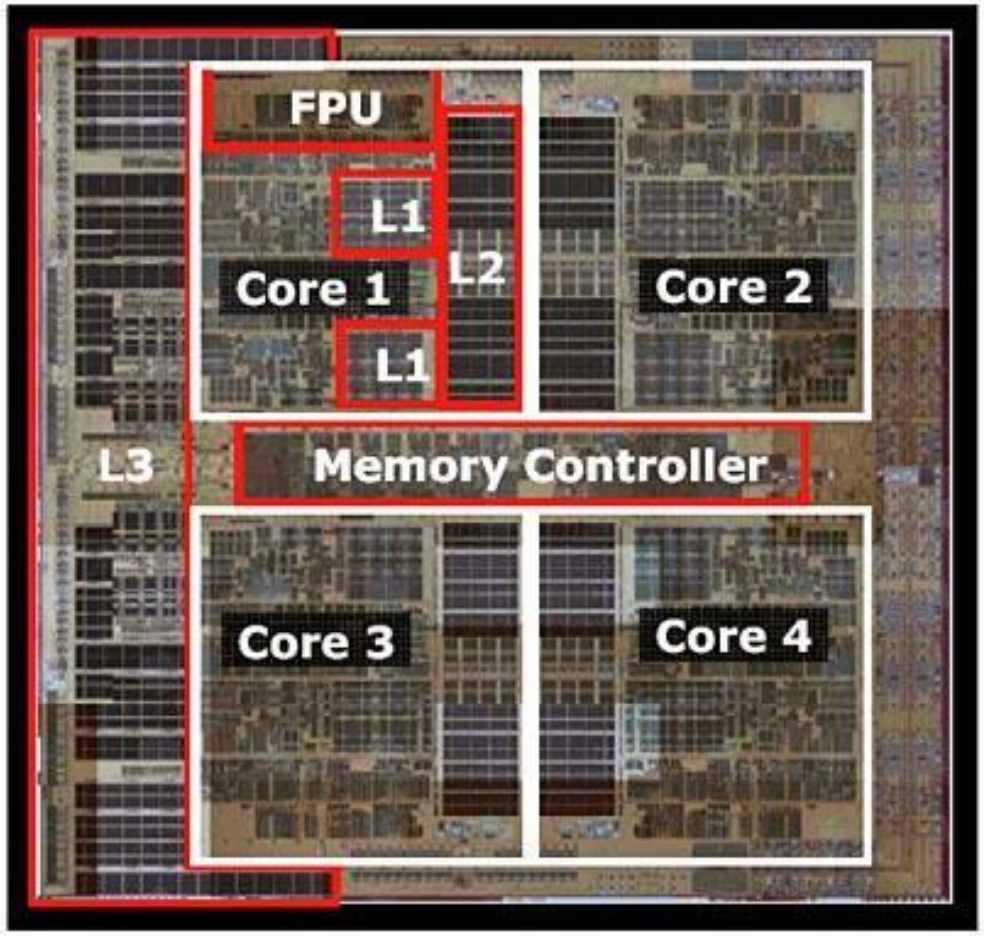
1. El procesador solicita contenidos de una posición (dirección) de memoria.
2. La cache comprueba si ya están los datos de esta posición:
   - **Si está (ACIERTO)**,
     - **3.A.1** Se la sirve al procesador desde la cache (rápidamente).
   - **Si no está (FALLO)**,
     - **3.B.1** La cache transfiere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

Ejemplo: AMD quad-core

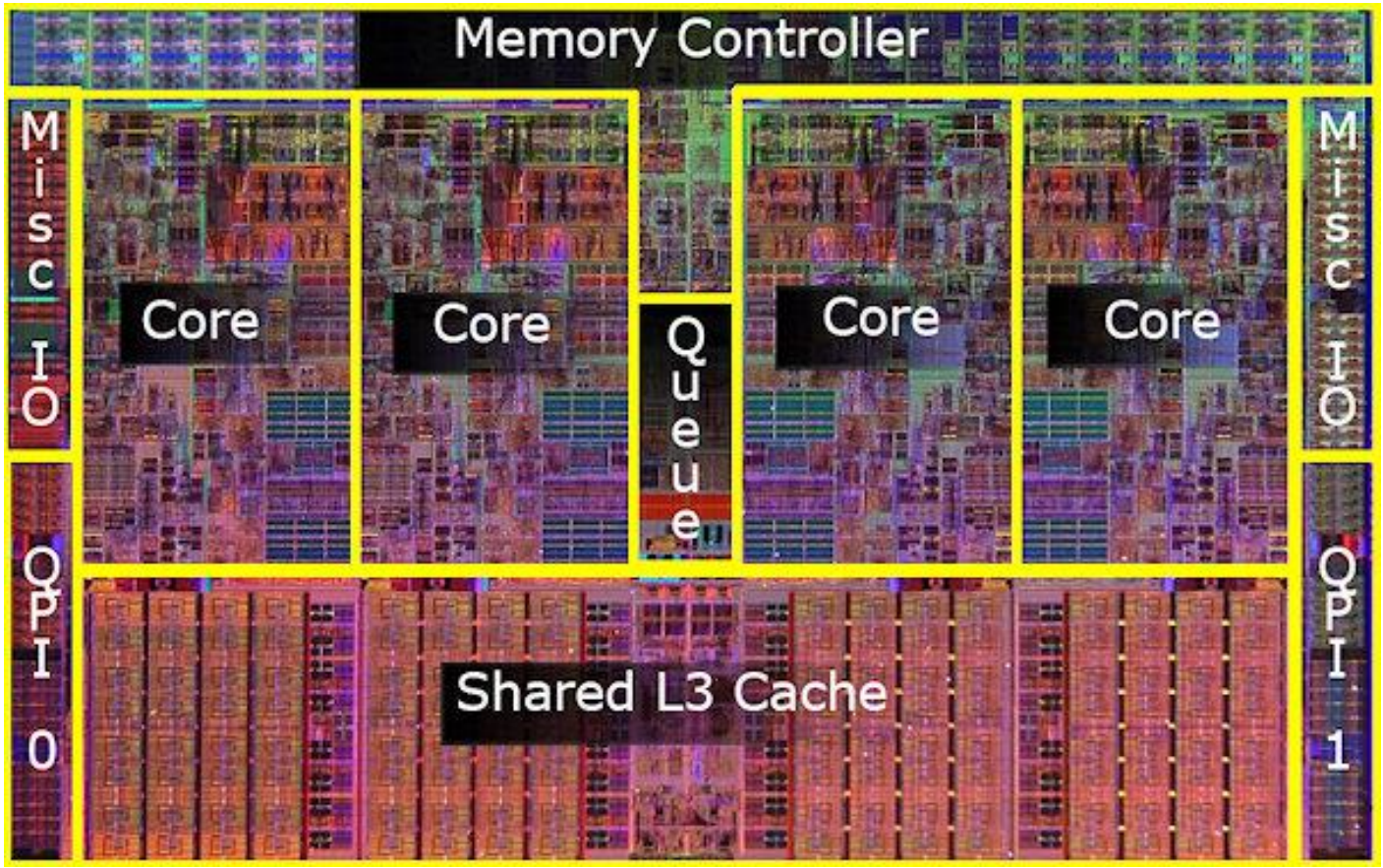


Quad-Core CPU mit gemeinsamen L3-Cache

Ejemplo: AMD quad-core



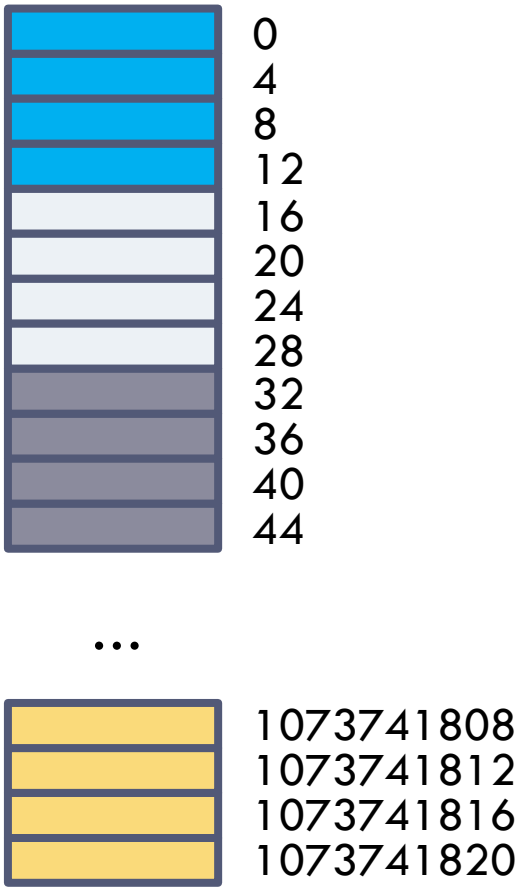
Ejemplo: Intel Core i7



Diseño y organización de la memoria caché

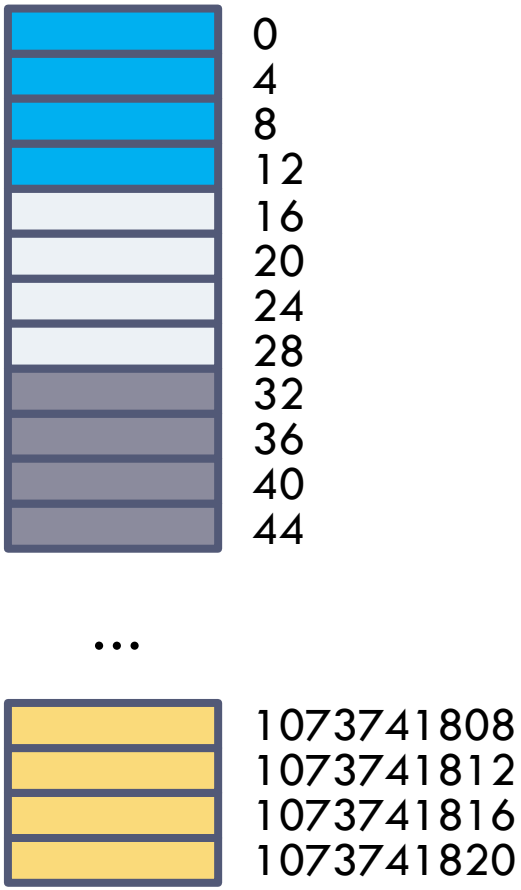
1. **Estructura de la memoria caché**
2. **Diseño de la memoria cache**
   - Tamaño
   - Función de correspondencia
   - Algoritmo de sustitución
   - Política de escritura

Organización de la memoria principal



- La memoria principal *lógicamente* se divide en bloques de igual tamaño.
  - 1 bloque = k palabras.
- Ejercicio: ¿Cuantos bloques de 4 palabras de 4 bytes 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.

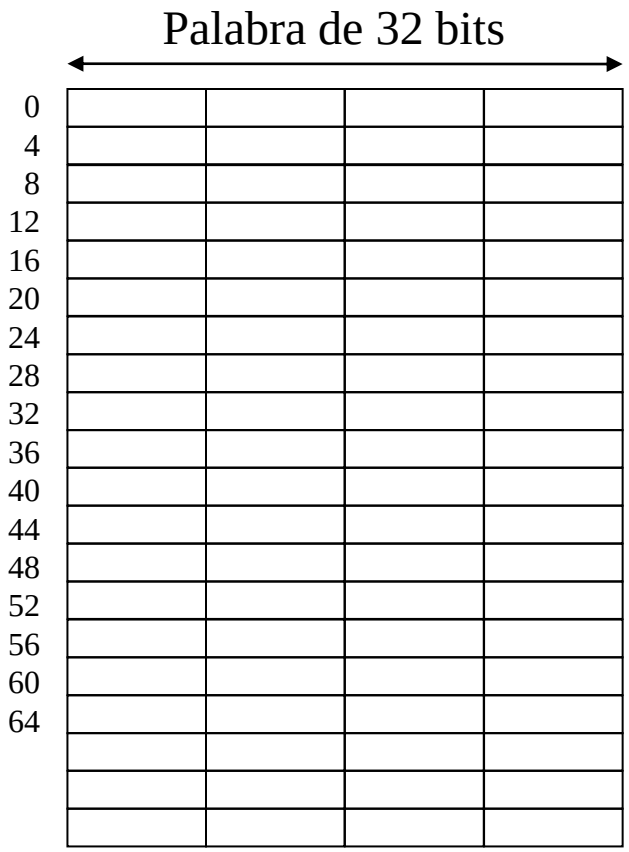
- Ejercicio: ¿Cuantos bloques de 4 palabras de 4 bytes hay en una memoria de 1 GB?
  - *Solución*:

    2^{30} B / 16 B = 2^{30-4} B = 2^{26} B = 64 megabloques ≈ 64 millones

Acceso a una palabra en memoria principal

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

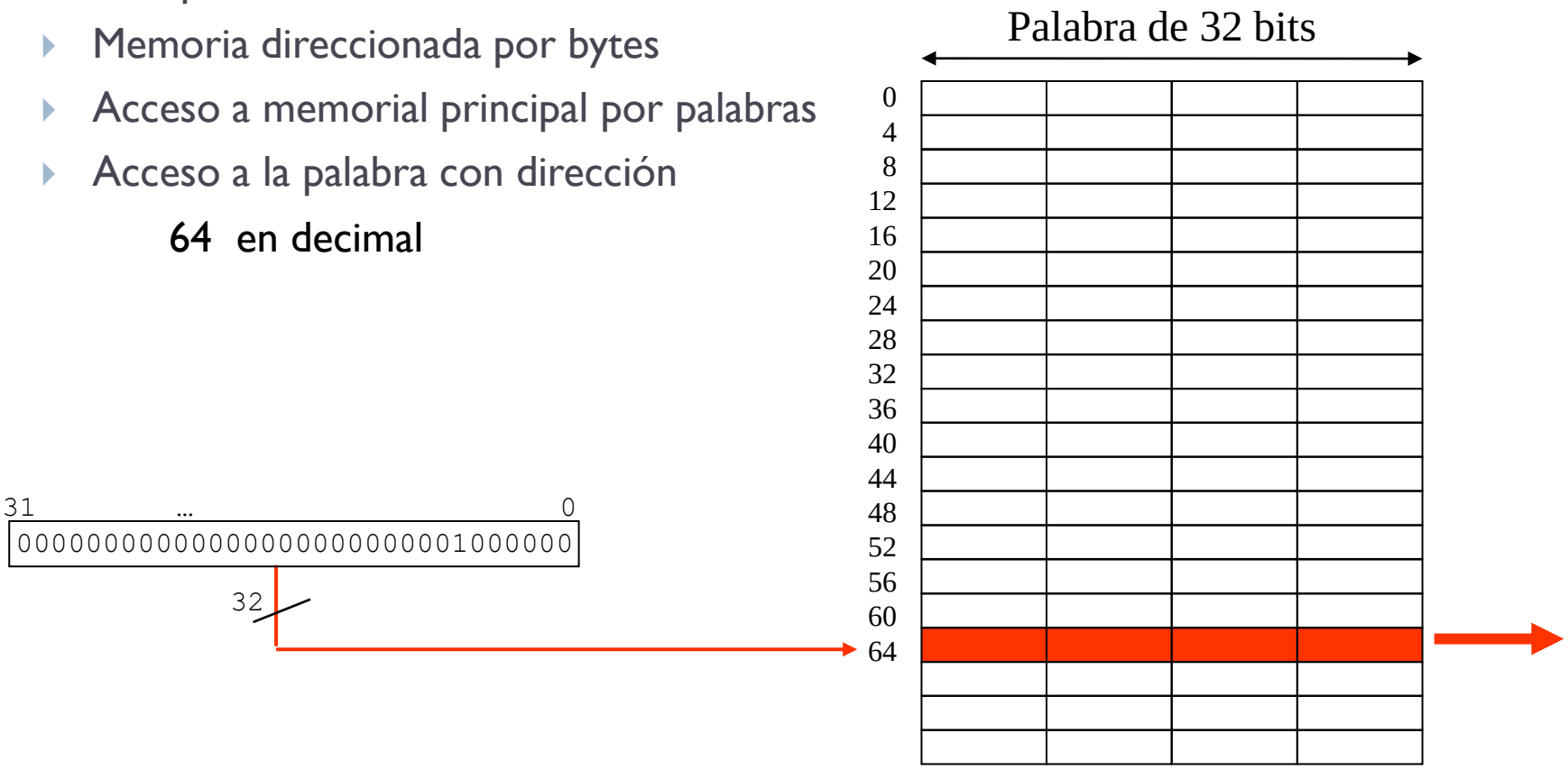
    64 en decimal



Acceso a una palabra en memoria principal

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

    64 en decimal



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

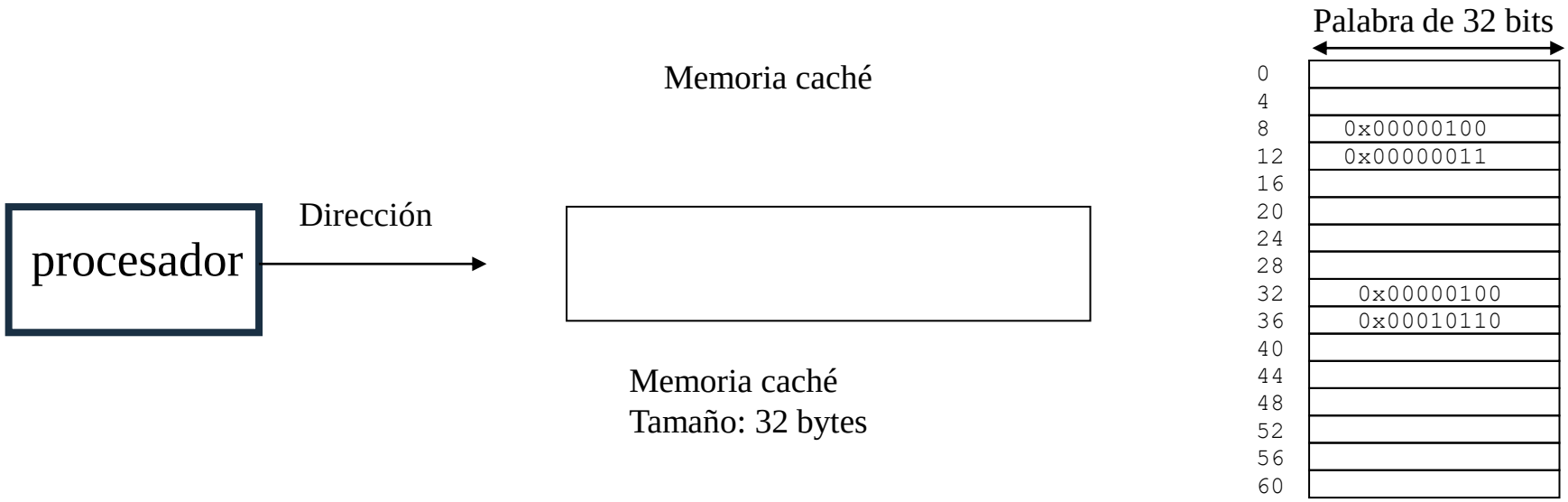
- La memoria caché está organizada en bloques (líneas)
  - Al bloque de memoria caché se le llama línea de caché
- El tamaño del bloque de M. principal es igual al tamaño de la línea.
  - Pero el tamaño de la memoria caché es mucho menor.
  - El número de bloques que cabe en la memoria caché es muy pequeño.

- ¿Cuántos bloques de 4 palabras cabe en una memoria caché de 32 KB?
  - Solución:

    $2^5 \cdot 2^{10}$ B / 2^4 B = 2^{11} bloques = 2048 bloques = 2048 líneas

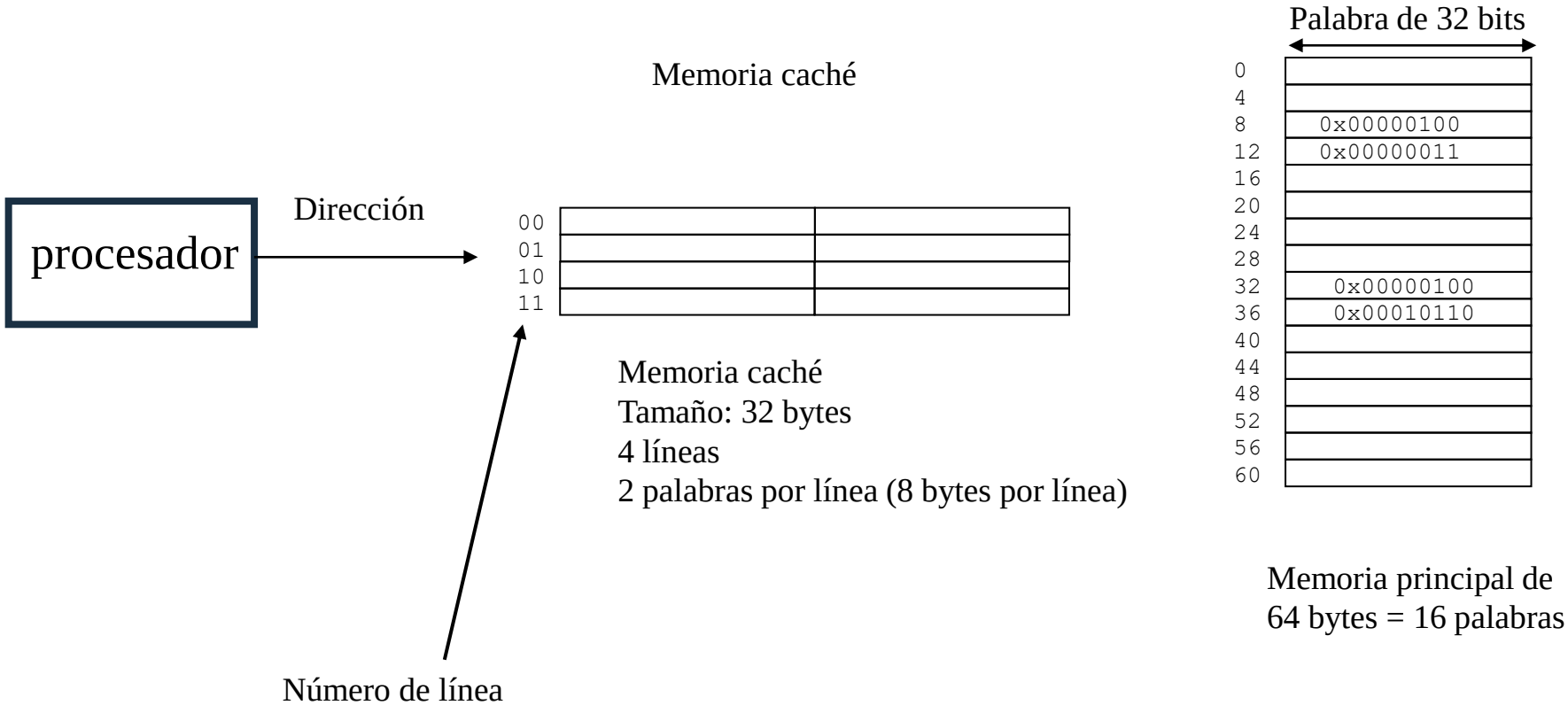
¿Cómo buscar una palabra en caché?

Ejemplo:

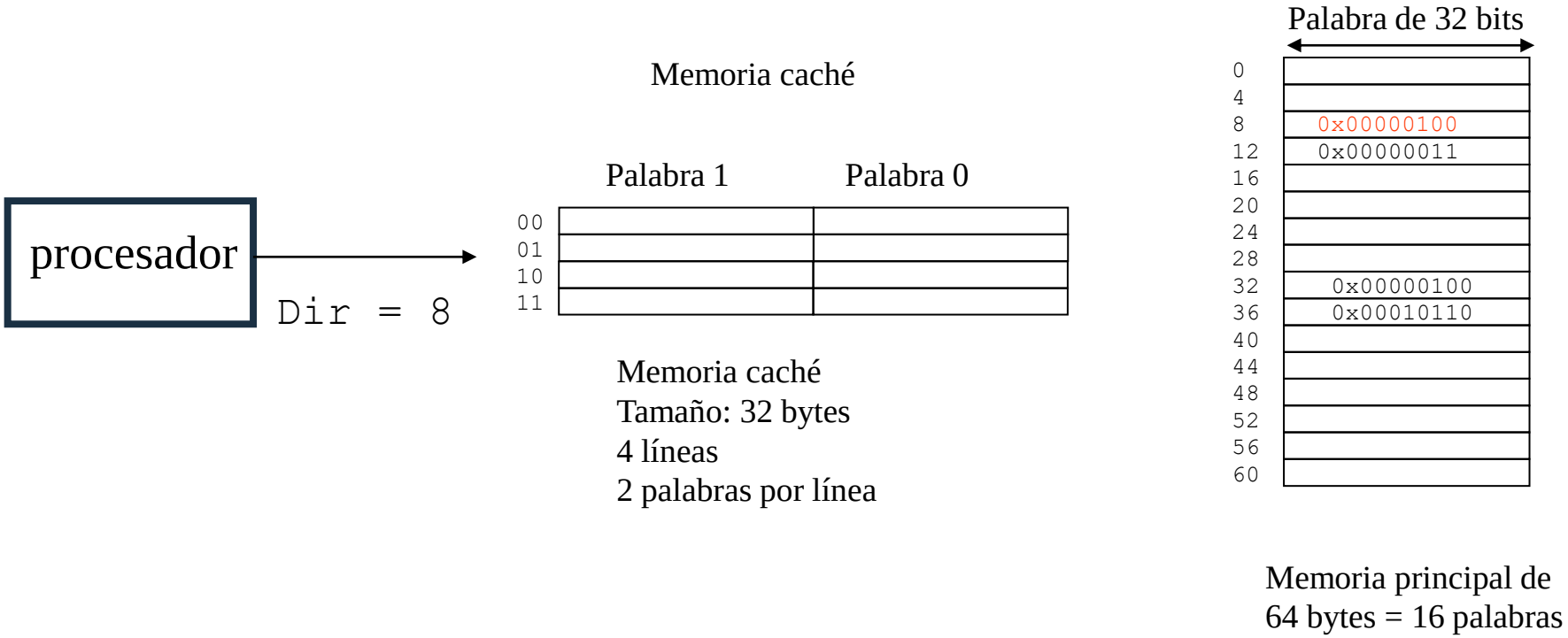


Con líneas de dos palabras
¿cuántas líneas tiene la caché?

¿Cómo buscar una palabra en cache?



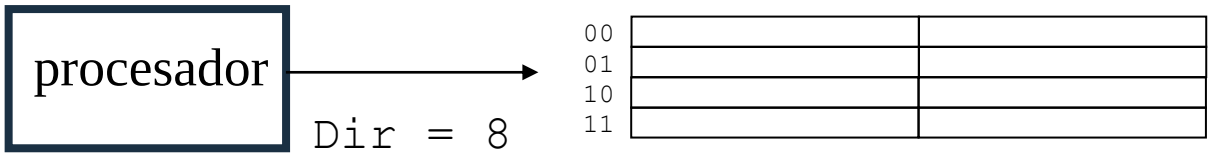
¿Cómo buscar una palabra en caché?



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

Memoria principal de
64 bytes = 16 palabras

¿Cómo buscar una palabra en caché?



Memoria caché

	Palabra 1	Palabra 0
00		
01		
10		
11		

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

FALLO

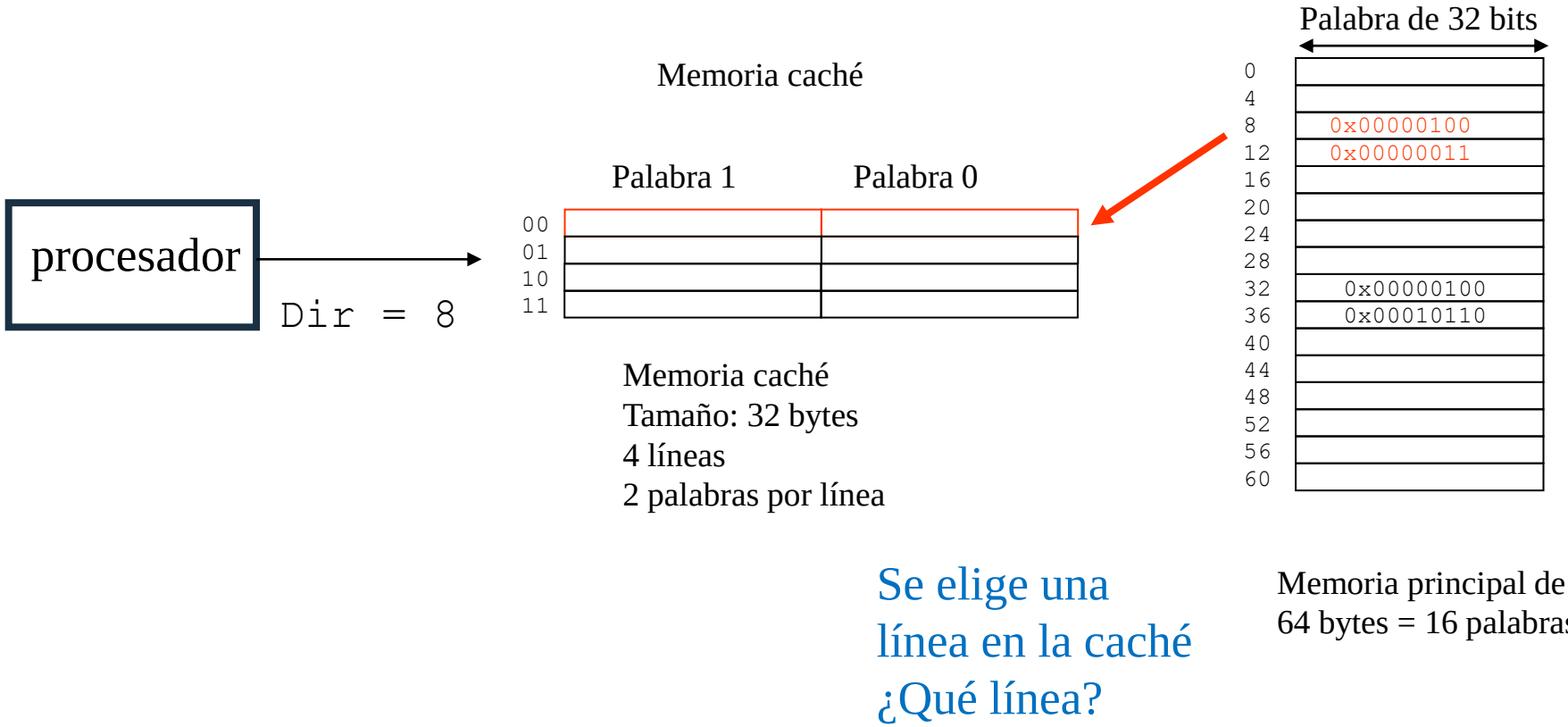
¿Cómo se sabe?

Palabra de 32 bits

0	
4	
8	0x00000100
12	0x00000011
16	
20	
24	
28	
32	0x00000100
36	0x00010110
40	
44	
48	
52	
56	
60	

Memoria principal de
64 bytes = 16 palabras

¿Cómo buscar una palabra en caché?

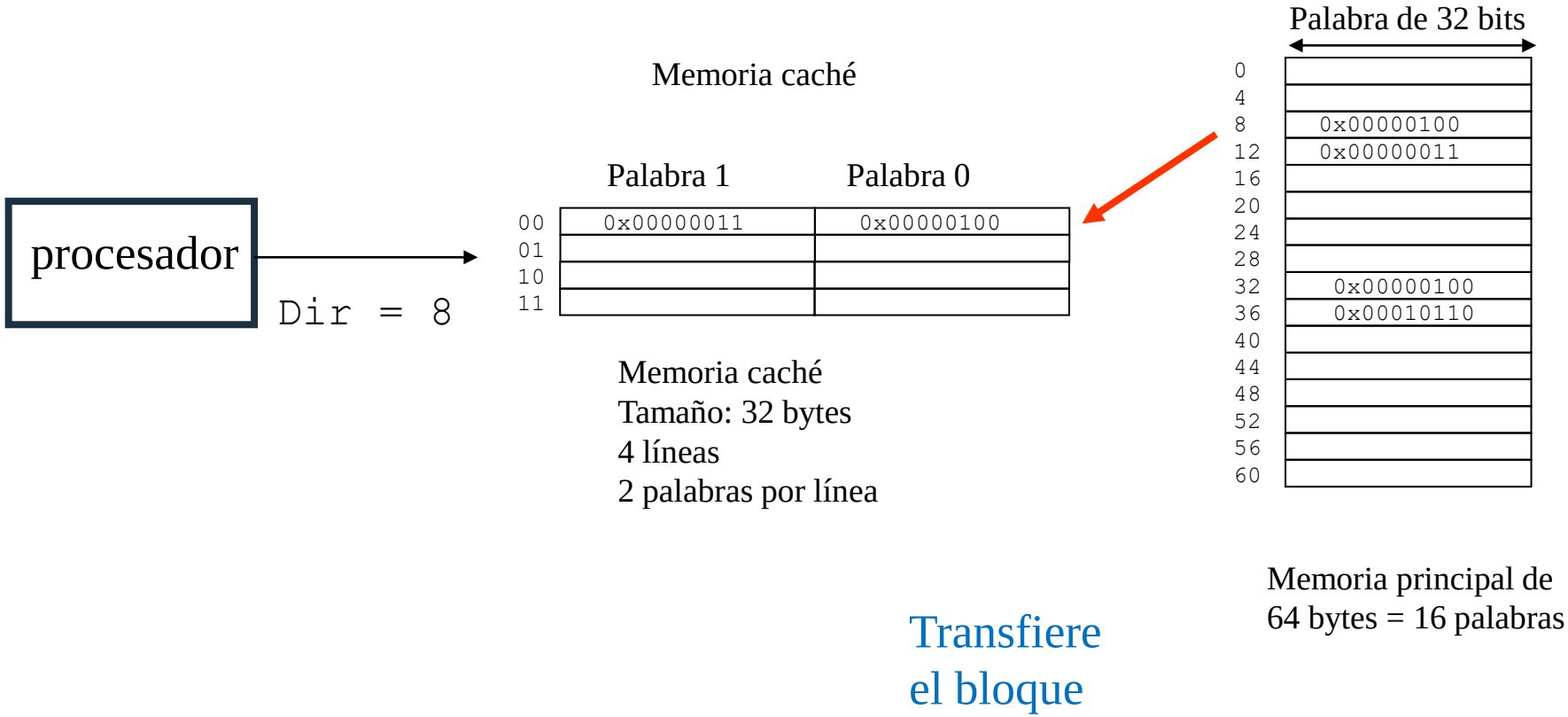


Memoria caché
Tamaño: 32 bytes
4 líneas
2 palabras por línea

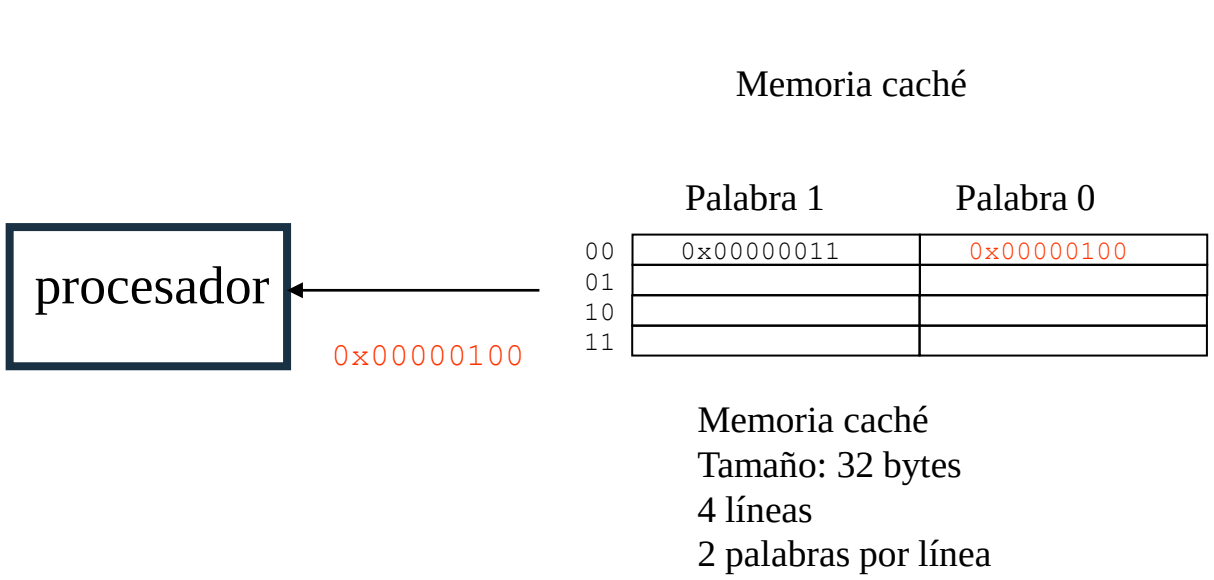
Se elige una línea en la caché ¿Qué línea?

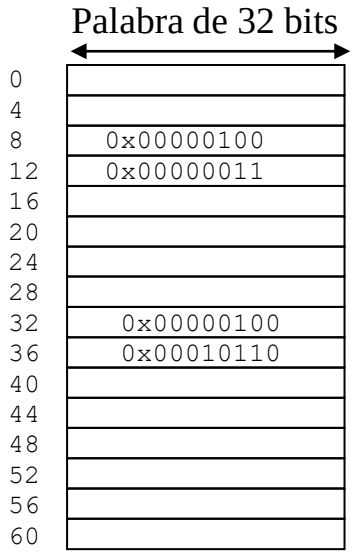
Memoria principal de 64 bytes = 16 palabras

¿Cómo buscar una palabra en caché?



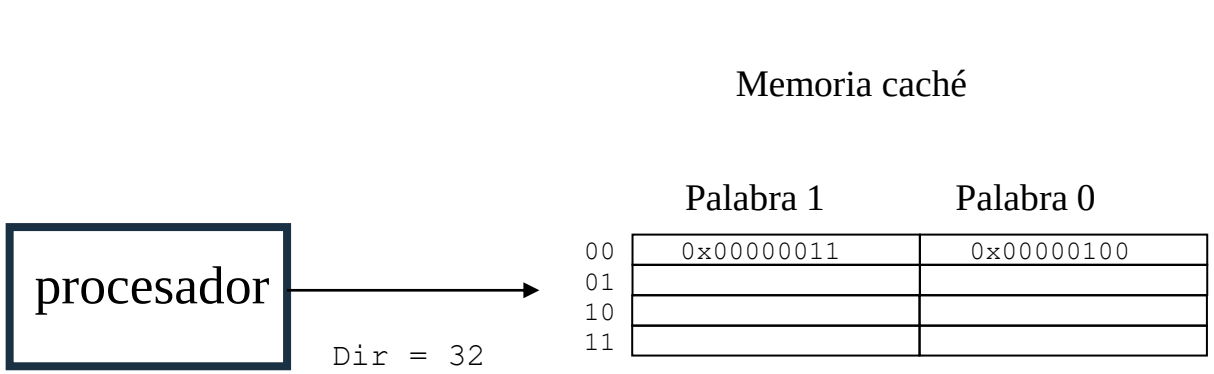
¿Cómo buscar una palabra en caché?



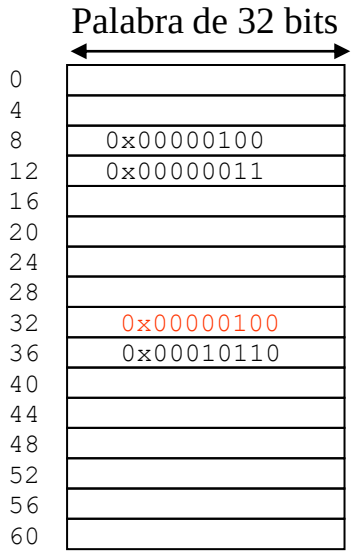


Transfiere
la palabra

¿Cómo buscar una palabra en caché?



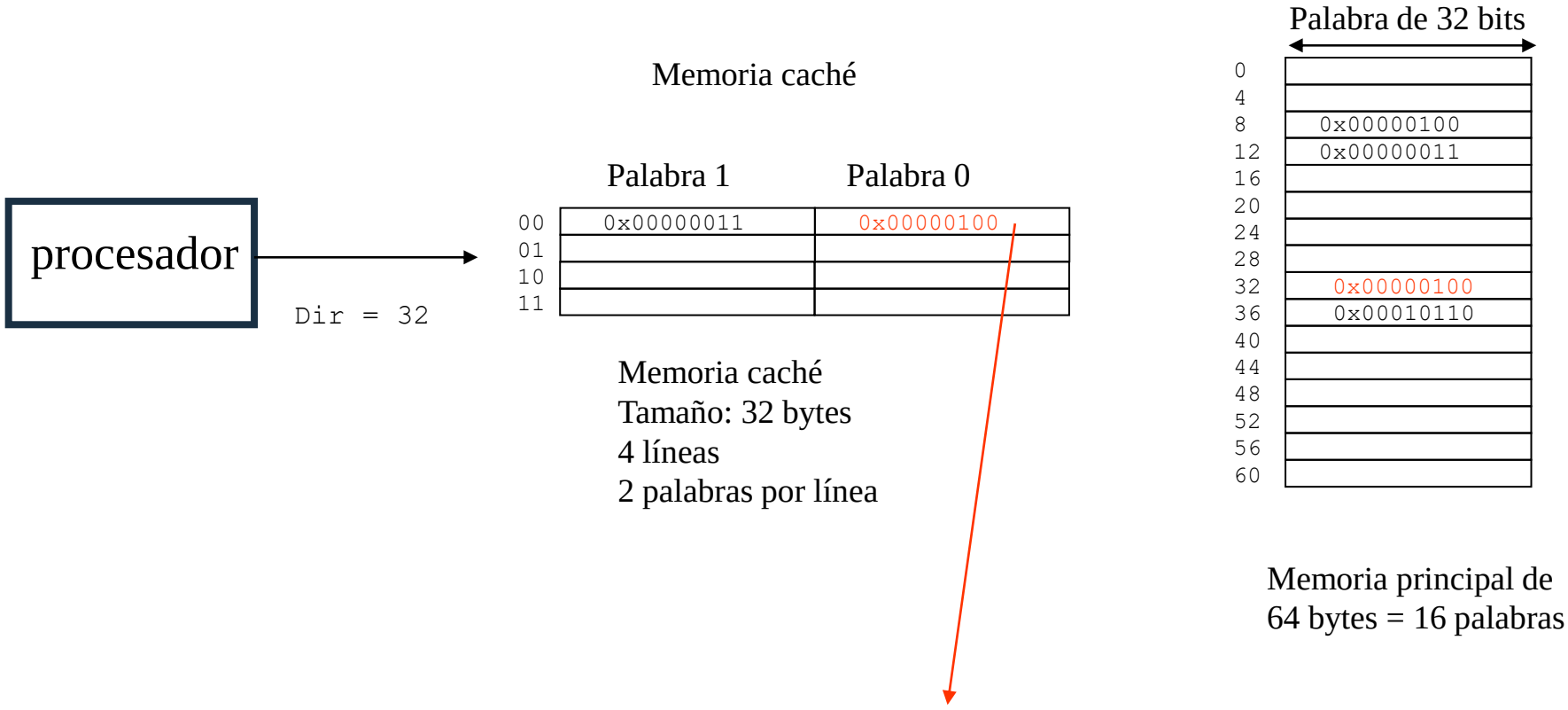
Memoria caché
Tamaño: 32 bytes
4 líneas
2 palabras por línea



Memoria principal de
64 bytes = 16 palabras

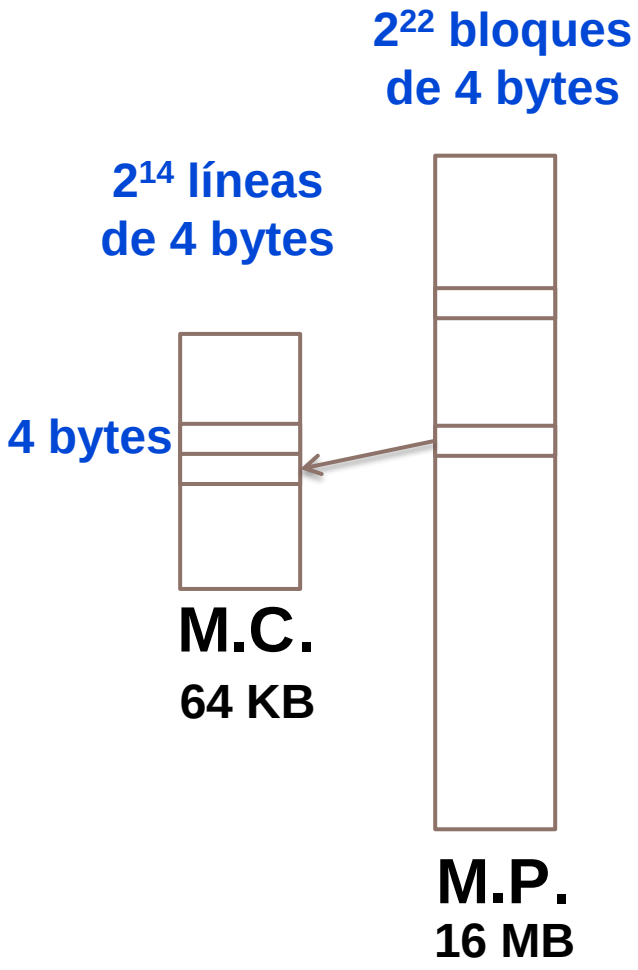
¿Cómo saber si está en la caché?

¿Cómo buscar una palabra en caché?



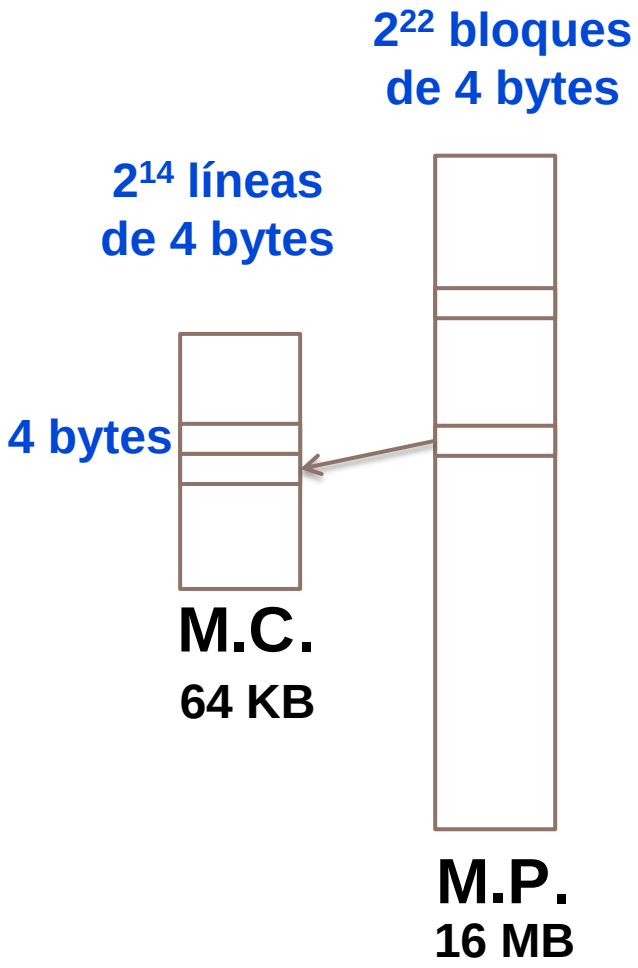
Es el contenido de la dirección 8 no de la 32

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)

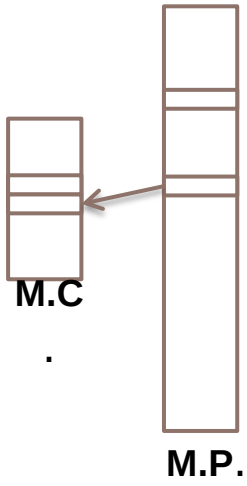
Estructura y diseño de la cache



- 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

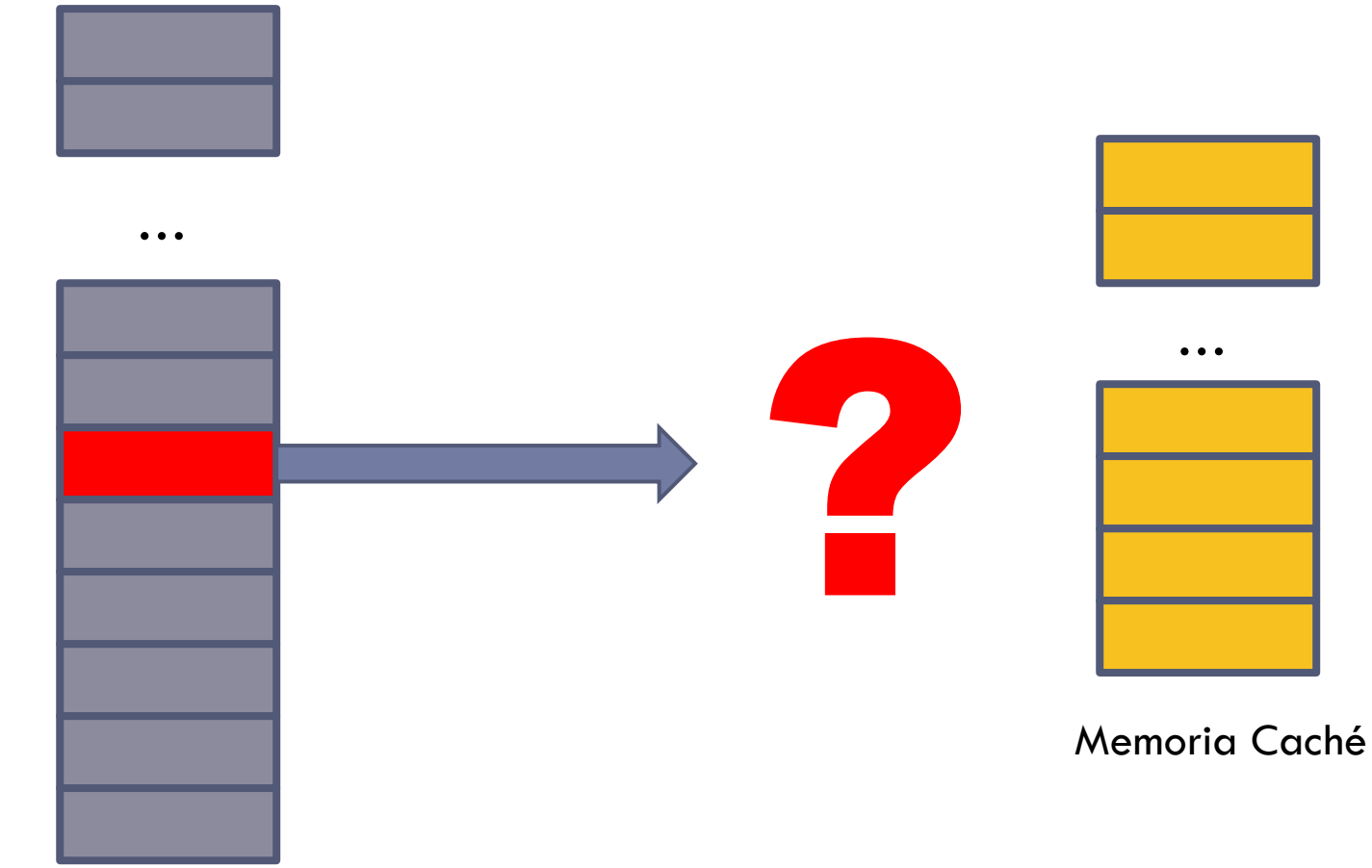


Función de correspondicena

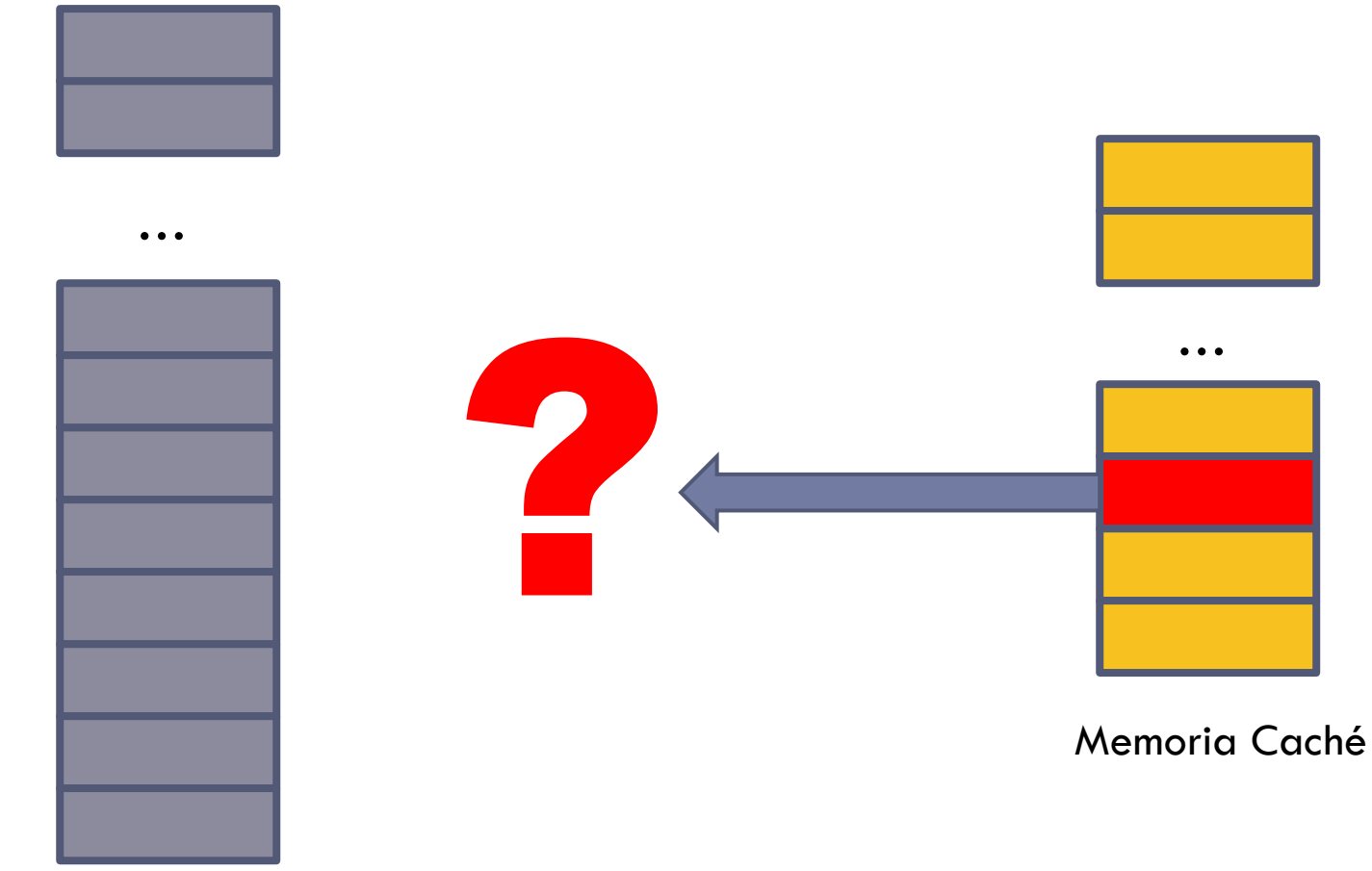
- Algoritmo que determina en qué lugares de la memoria caché se puede almacenar un bloque concreto de la memoria principal

- 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

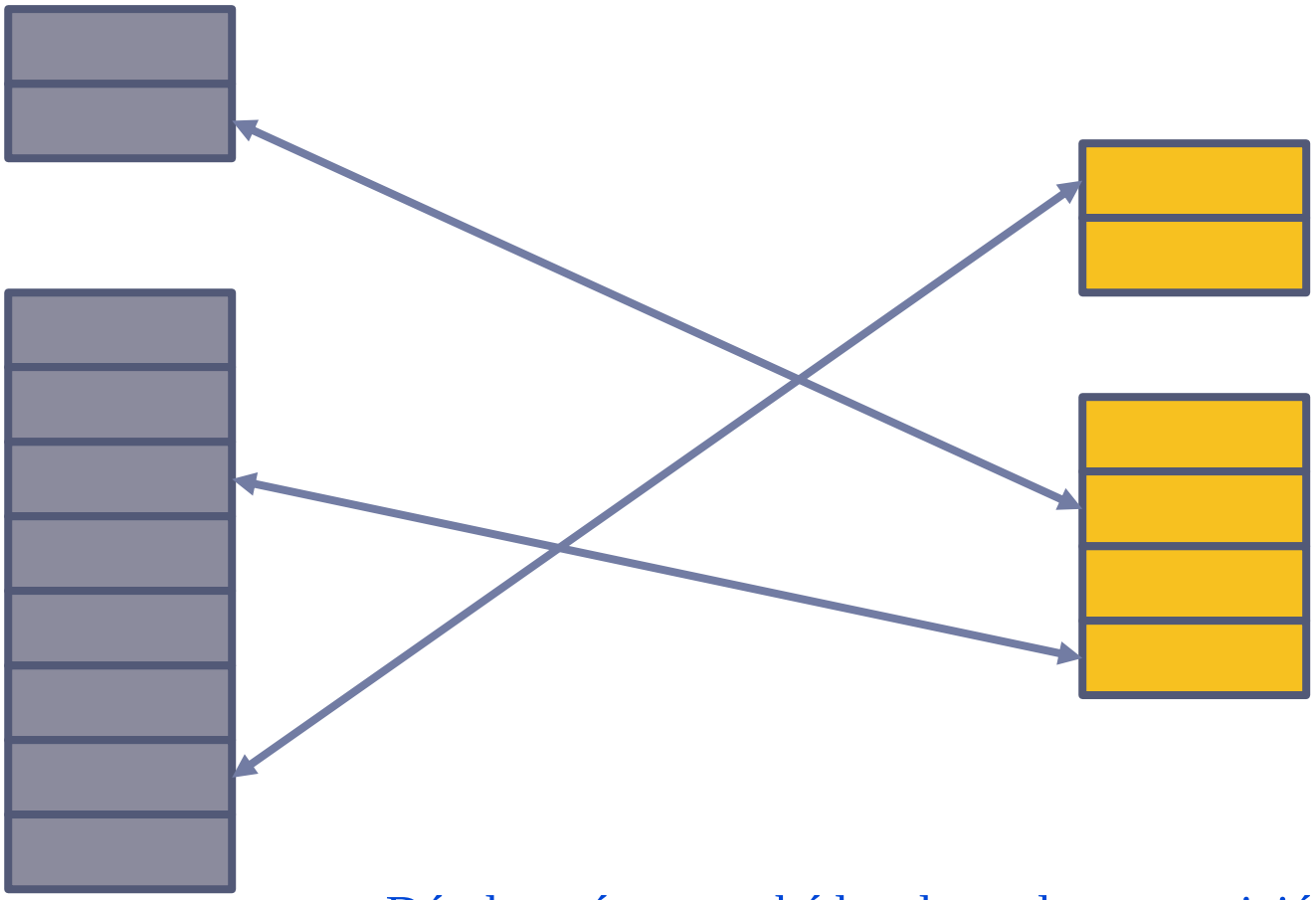
Ubicación en caché



Ubicación en memoria principal



Función de correspondencia

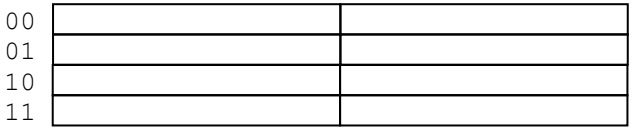


¿Dónde están en caché los datos de una posición de M.P.?

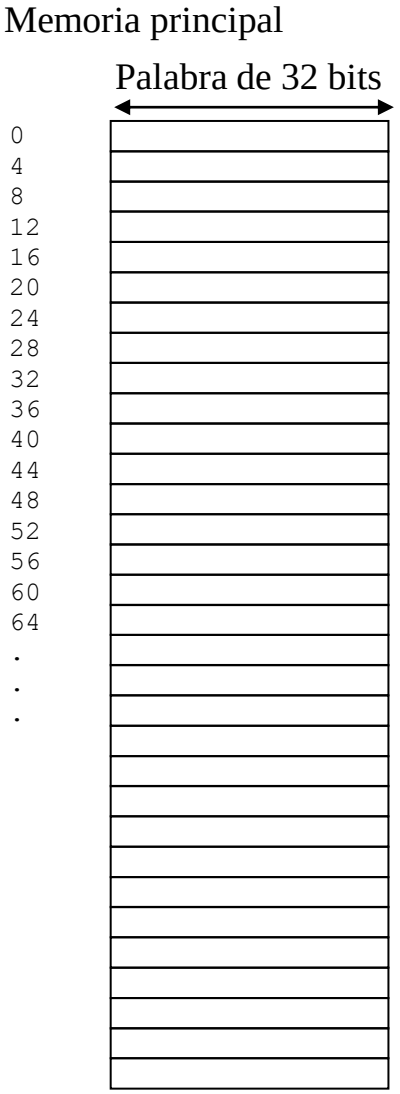
Funciones de correspondencia

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

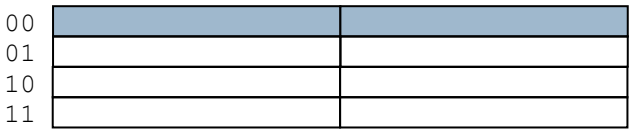
Correspondencia directa



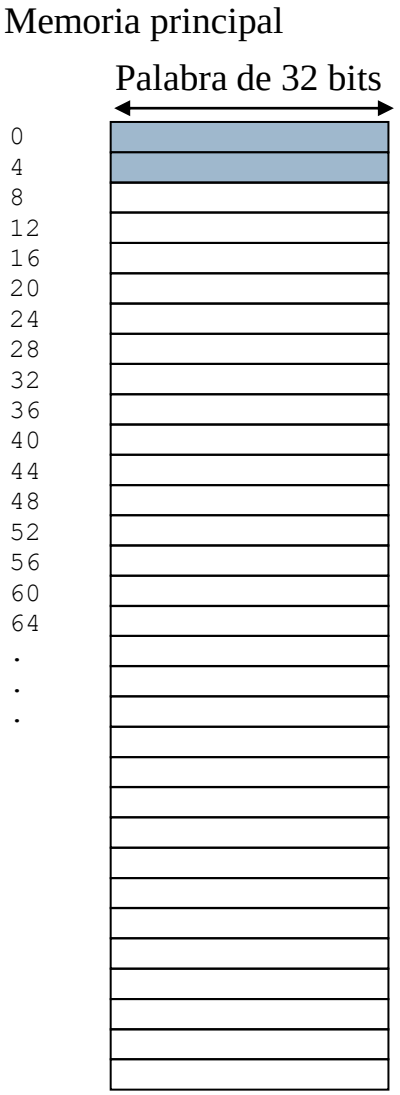
Memoria caché
Tamaño: 32 bytes
4 líneas
2 palabras por línea



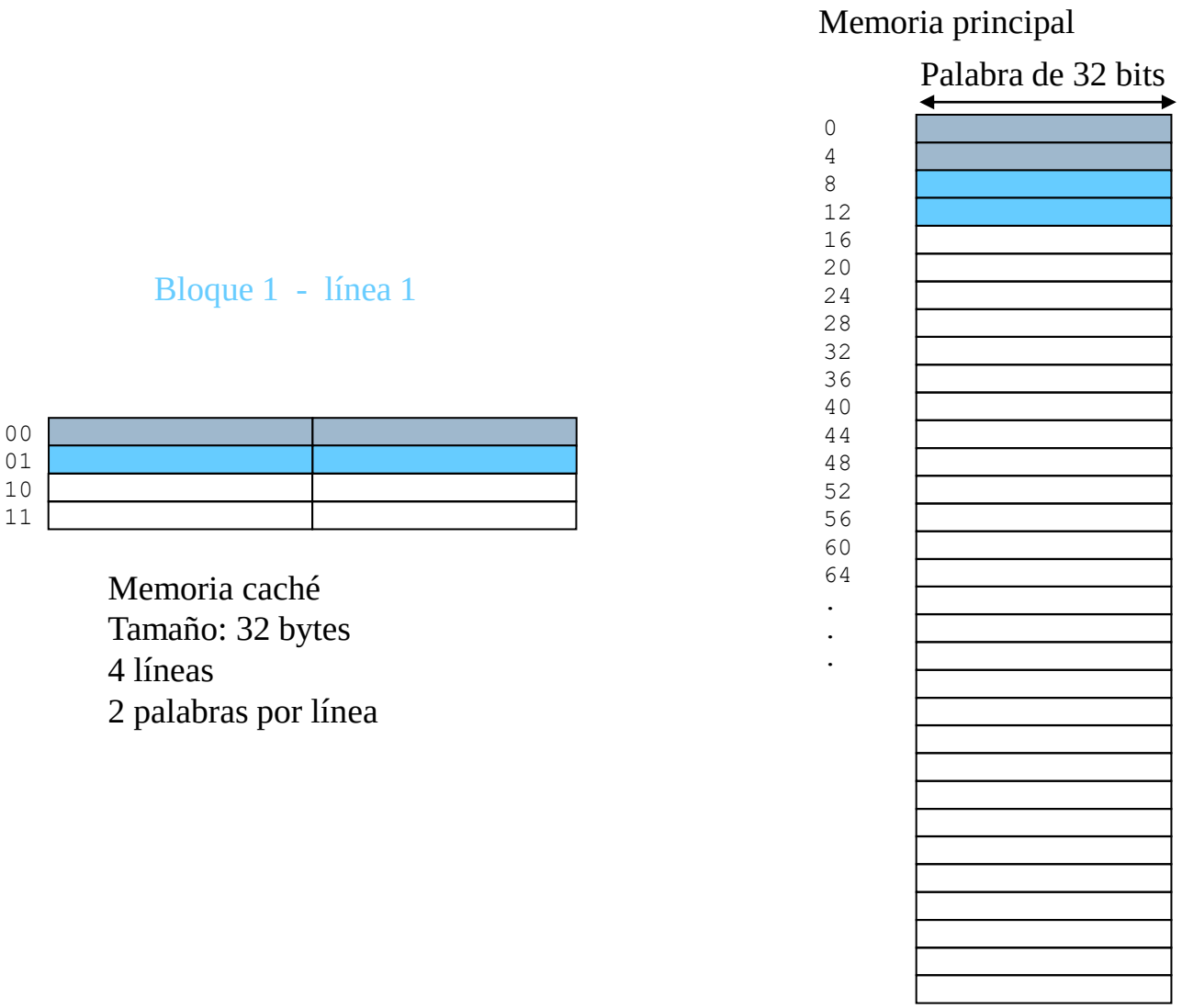
Correspondencia directa



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

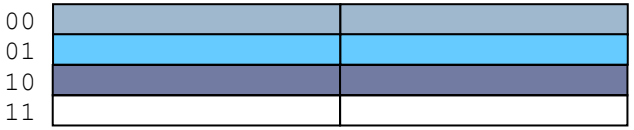


Correspondencia directa



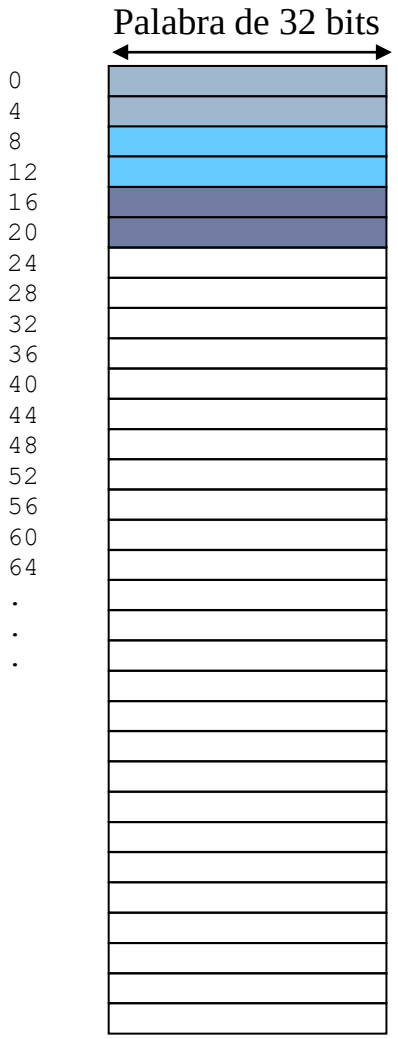
Correspondencia directa

Bloque 2 - línea 2

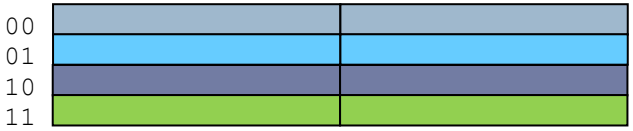


Memoria caché
Tamaño: 32 bytes
4 líneas
2 palabras por línea

Memoria principal

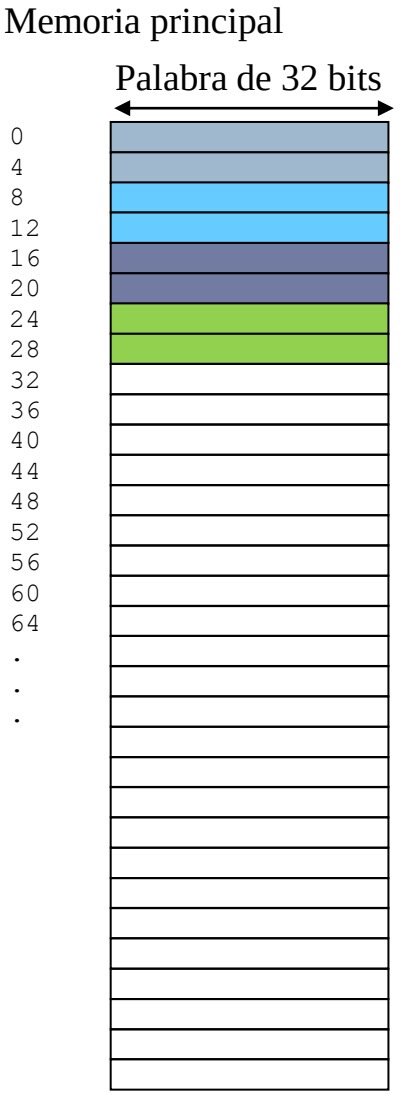


Correspondencia directa

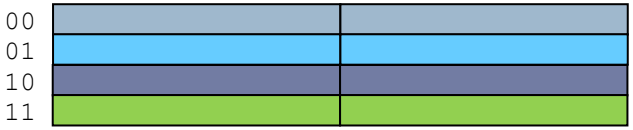


Bloque 3 - línea 3

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

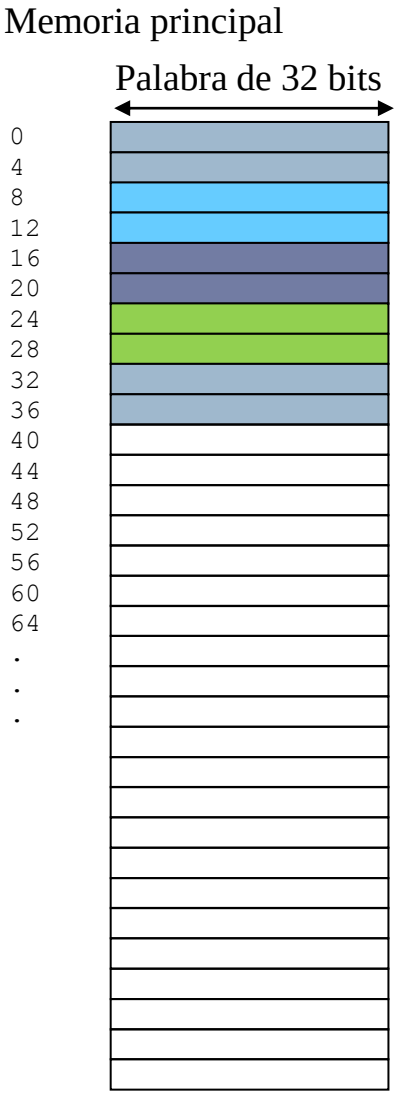


Correspondencia directa



Bloque 4 - línea 0

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

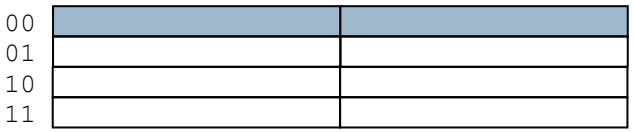


Correspondencia directa

- En general:
  - El bloque de memoria K se almacena en la línea:

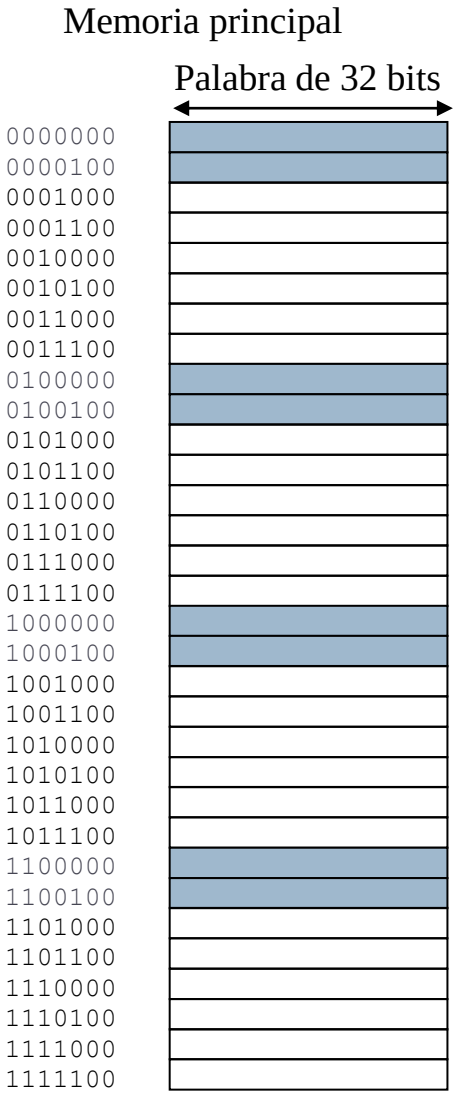
    K mod número de líneas

Correspondencia directa

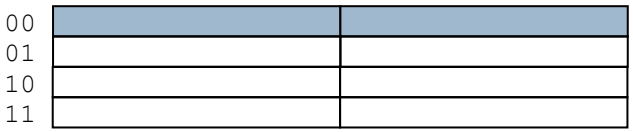


Memoria caché
Tamaño: 32 bytes
4 líneas
2 palabras por línea

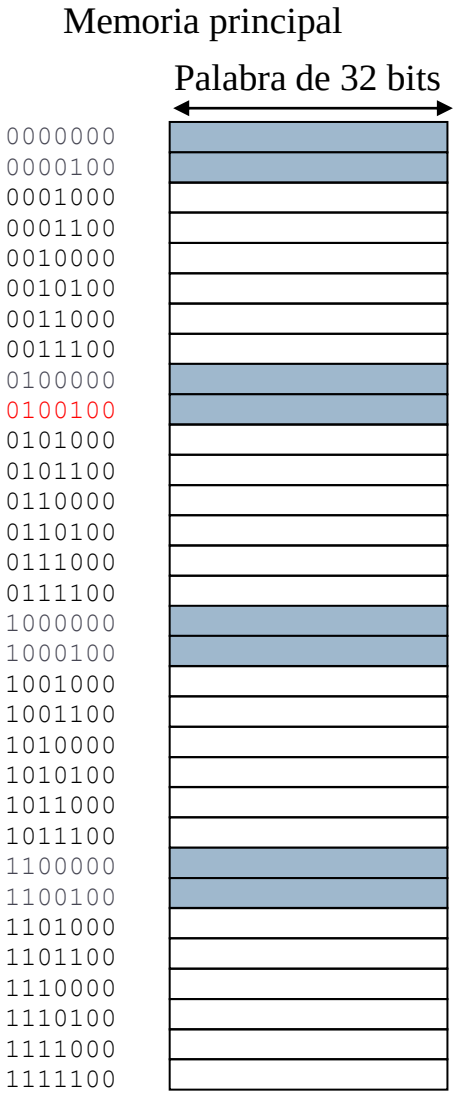
Varios bloques en la misma línea



Correspondencia directa

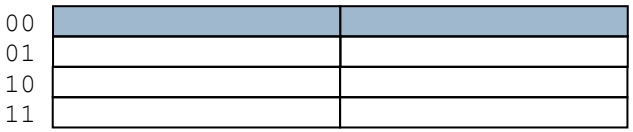


Memoria caché
Tamaño: 32 bytes
4 líneas
2 palabras por línea

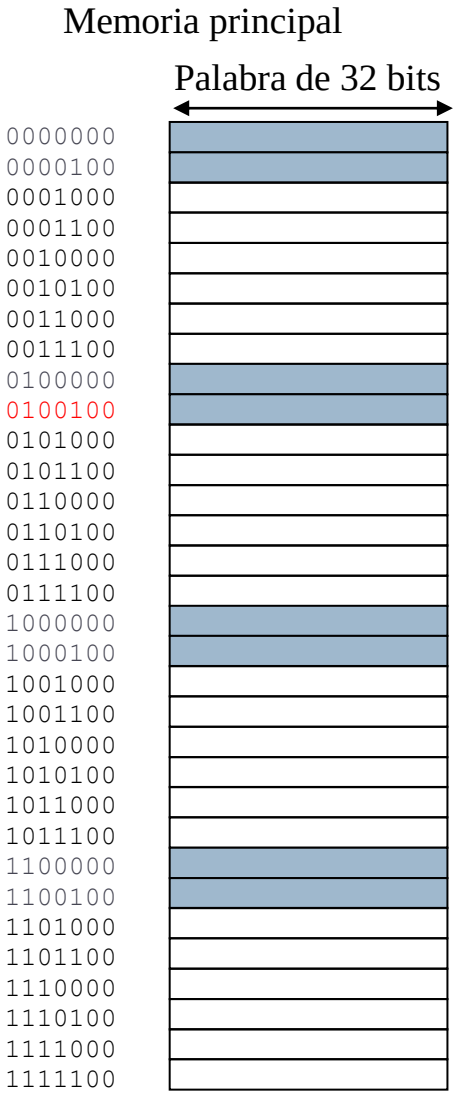


¿Cómo se sabe qué bloque de memoria se encuentra una determinada línea?
Ejemplo: la dirección 0100100

Correspondencia directa



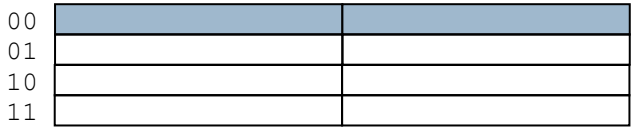
Memoria caché
Tamaño: 32 bytes
4 líneas
2 palabras por línea



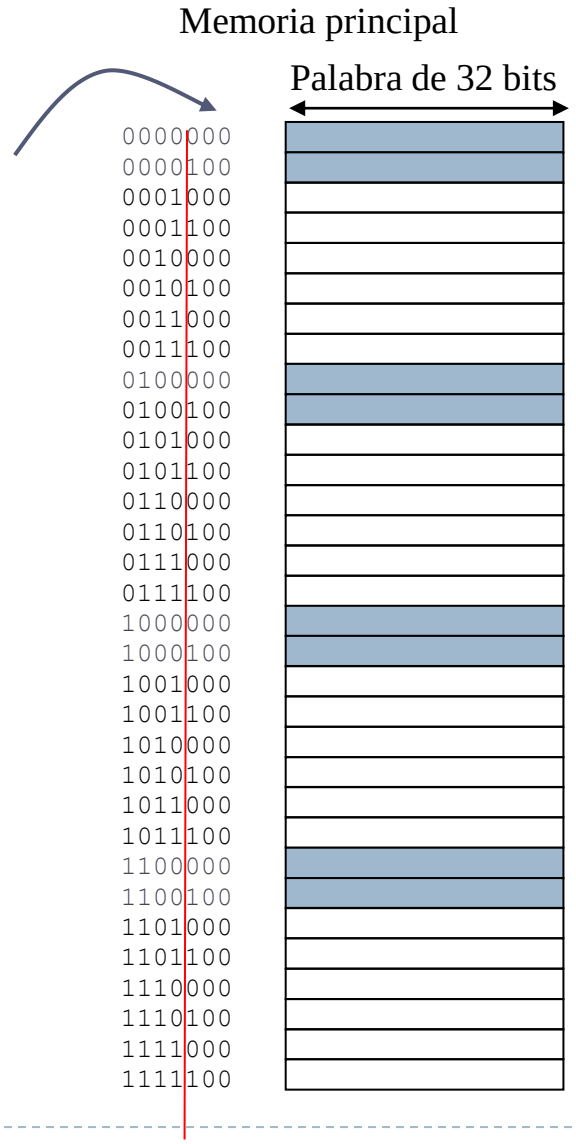
¿Cómo se sabe qué bloque de memoria se encuentra una determinada línea?
Ejemplo: la dirección 0100100
Se añade a cada línea una etiqueta

Correspondencia directa
Idea

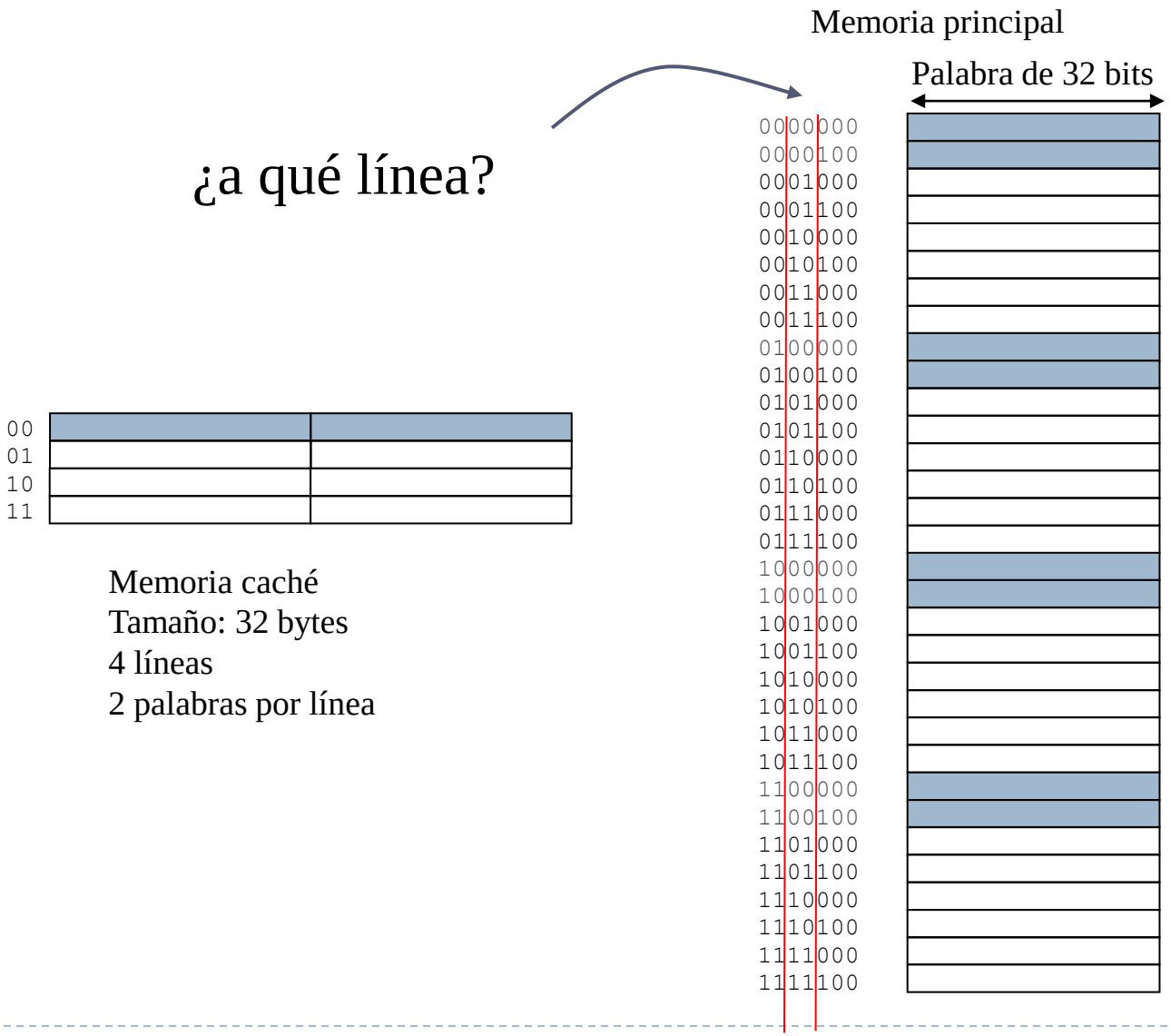
¿qué byte dentro de la línea?
Líneas de 8 bytes



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

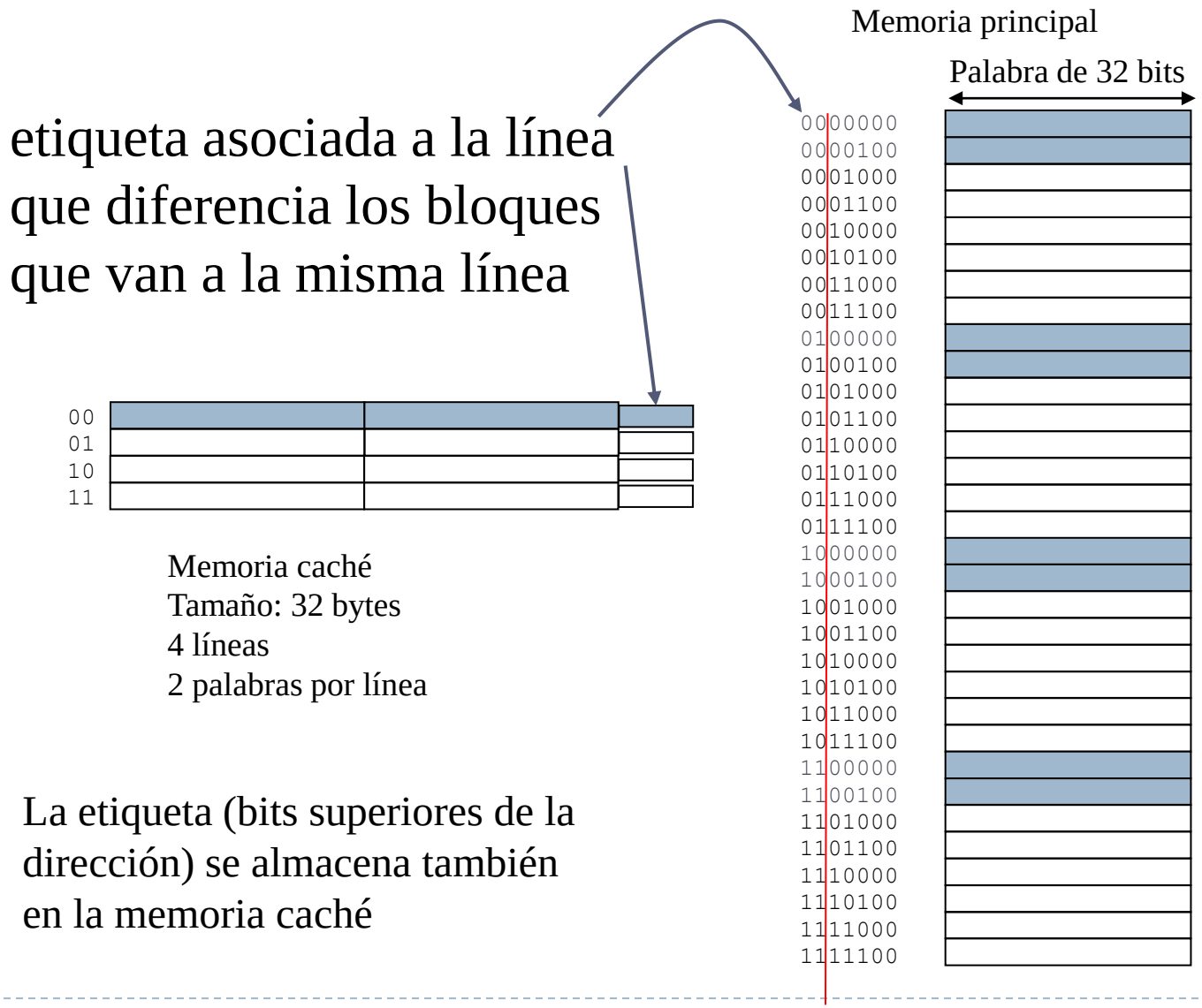


Correspondencia directa
Idea

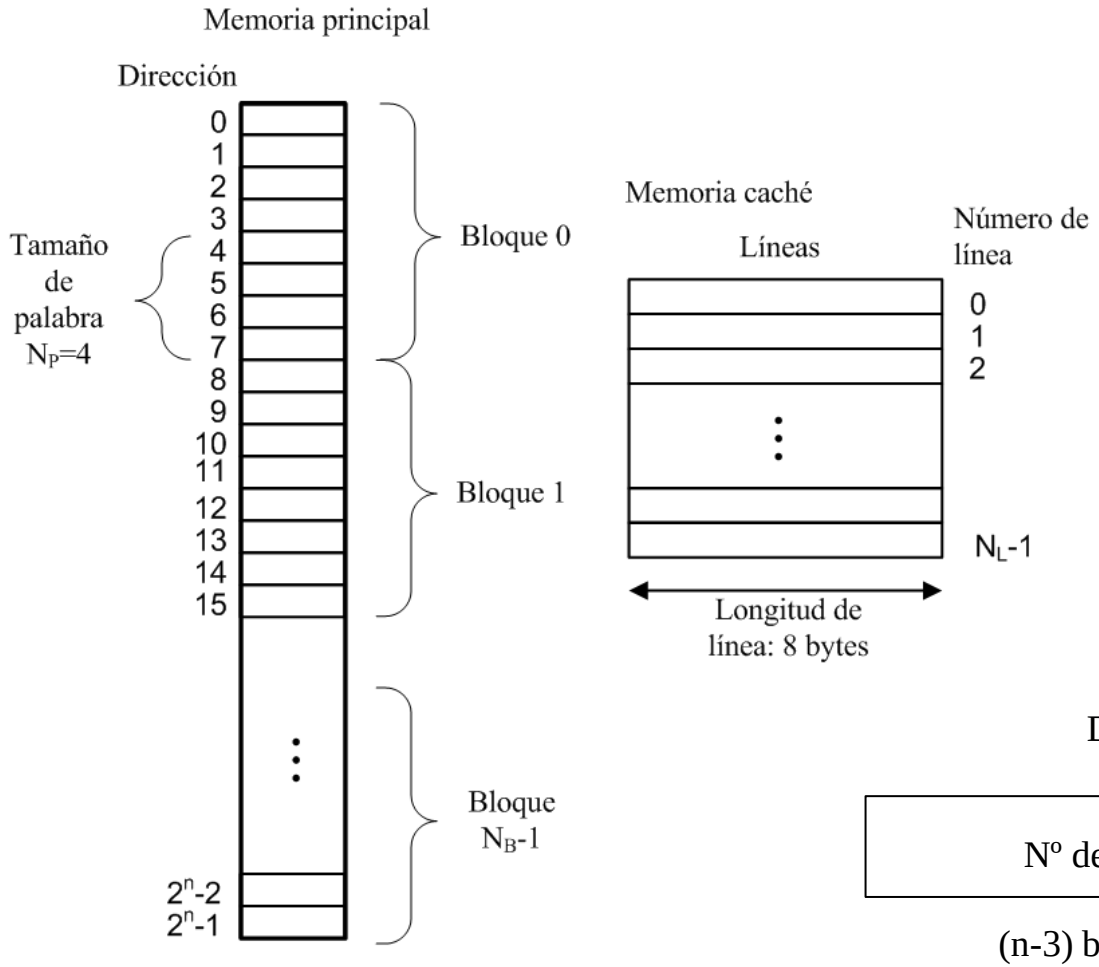


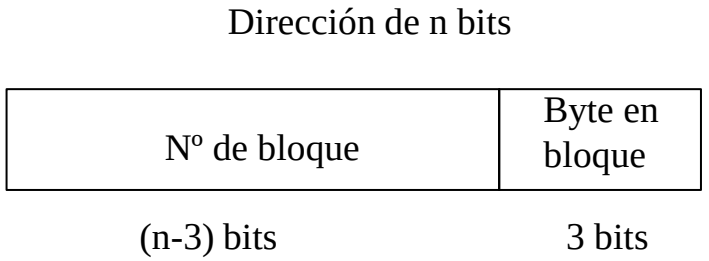
Memoria caché
Tamaño: 32 bytes
4 líneas
2 palabras por línea

Correspondencia directa
Idea

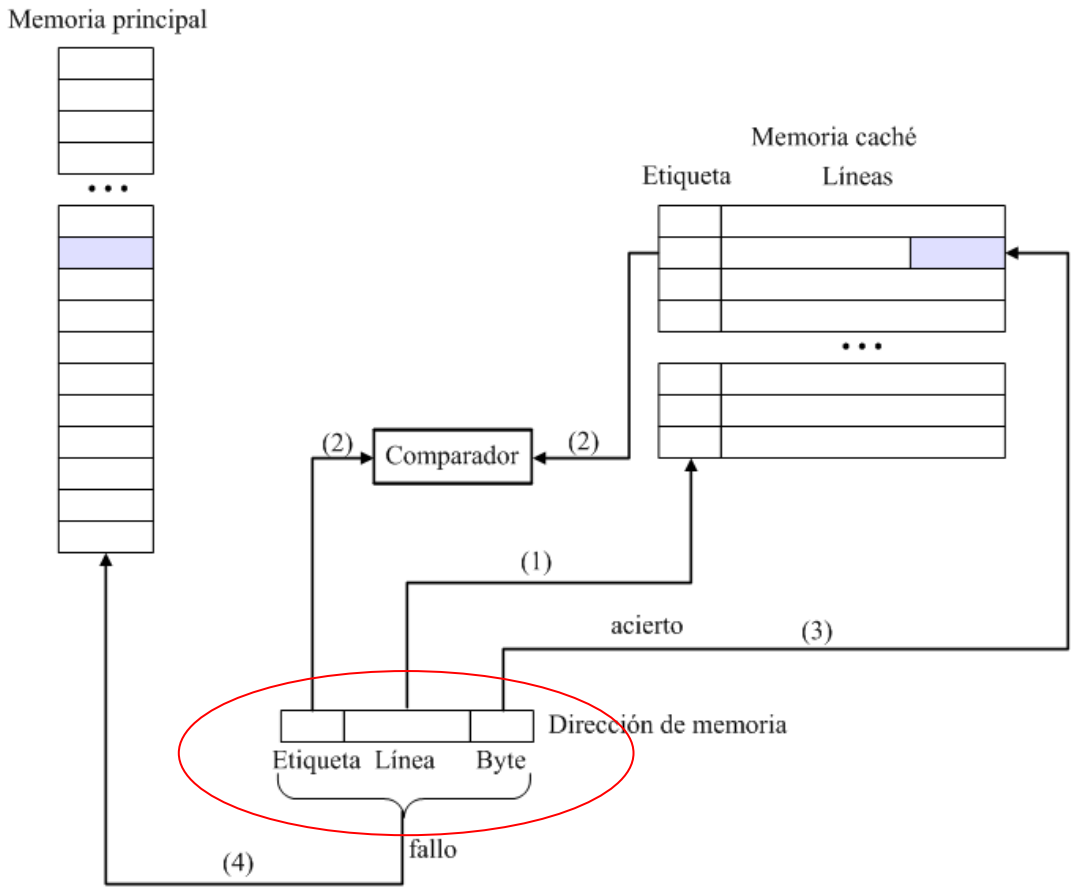


Ejemplo de organización de la memoria caché





Organización de una memoria caché con correspondencia directa

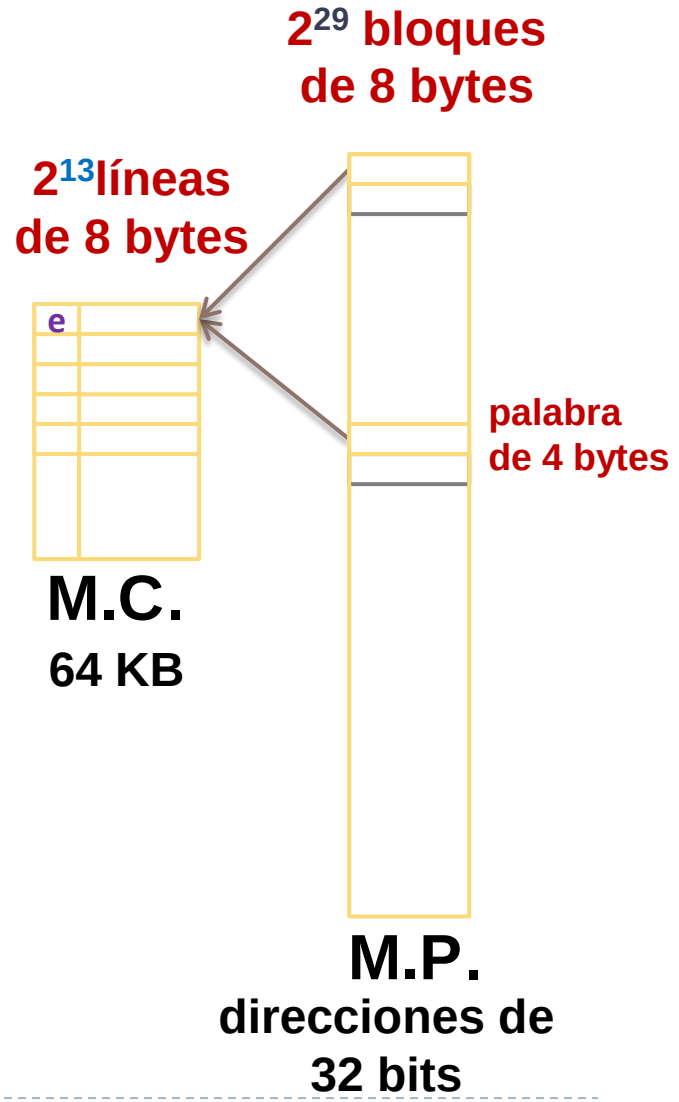


Función de correspondencia directa Ejemplo

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

32-16	13	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

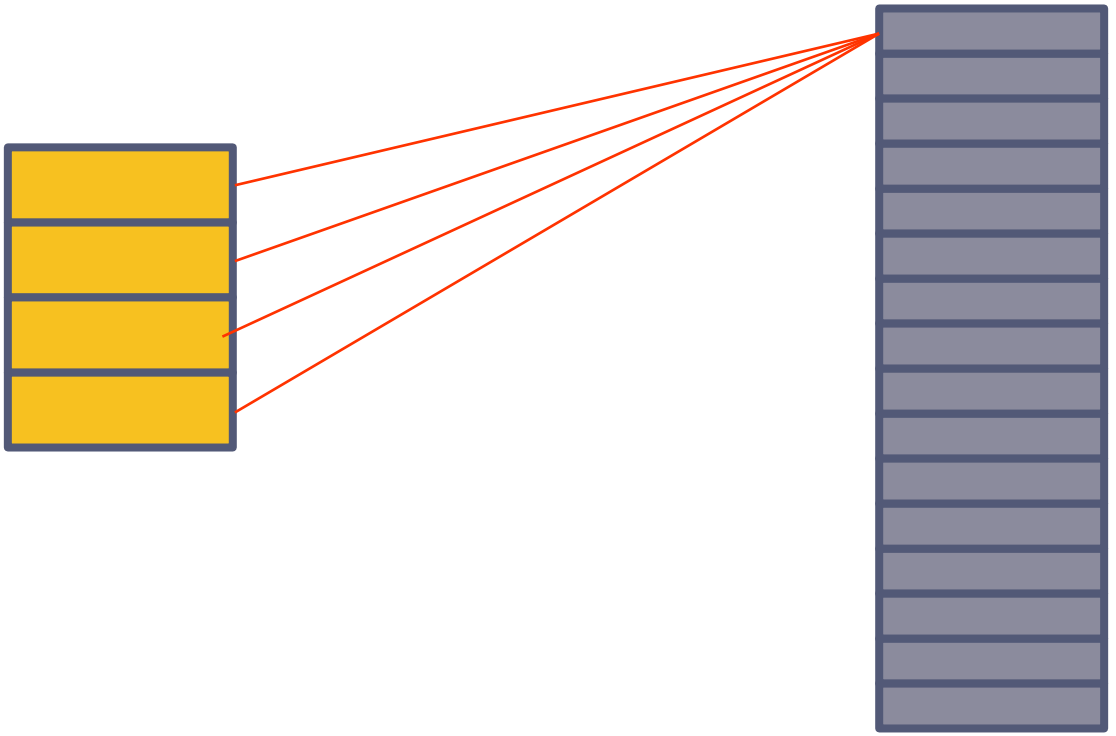


Ejercicio

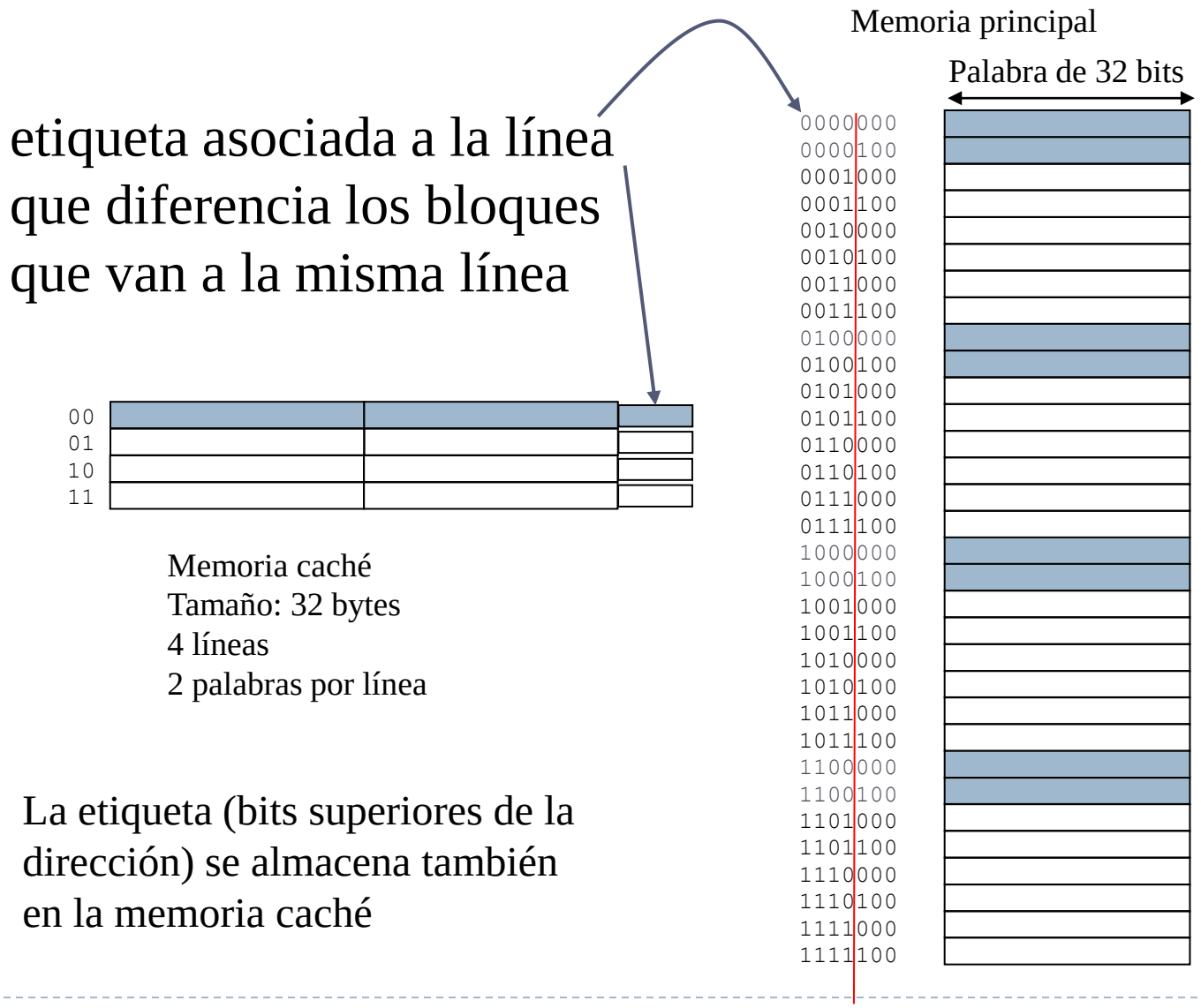
- Dado un computador de 32 bits con una memoria caché de 64 KB y bloques de 32 bytes. Si se utiliza correspondencia directa
  - ¿En qué línea de la memoria caché se almacena la palabra de la dirección 0x0000408A?
  - ¿Cómo se puede obtener rápidamente?
  - ¿En qué línea de la memoria caché se almacena 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

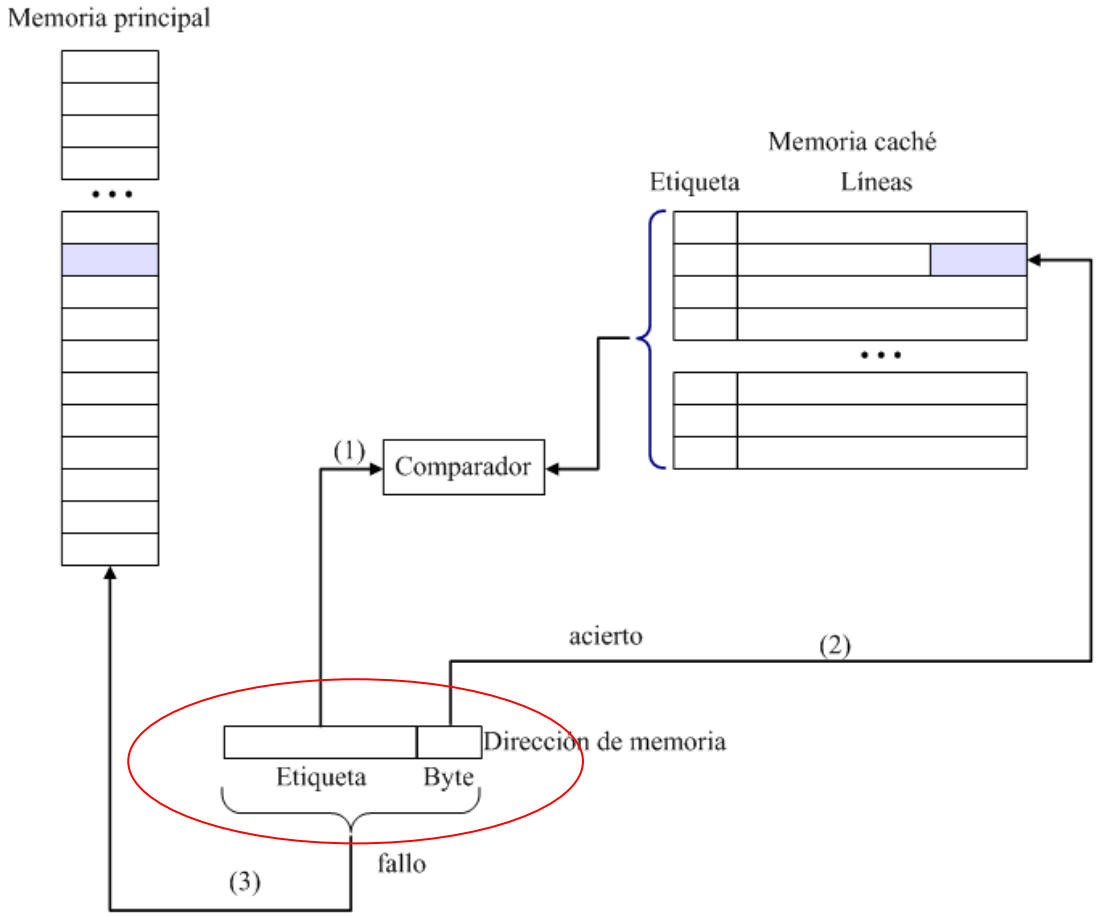
- Cada bloque de MP puede almacenarse en cualquier línea de la caché



Correspondencia asociativa
Idea



Organización de una memoria caché con correspondencia asociativa

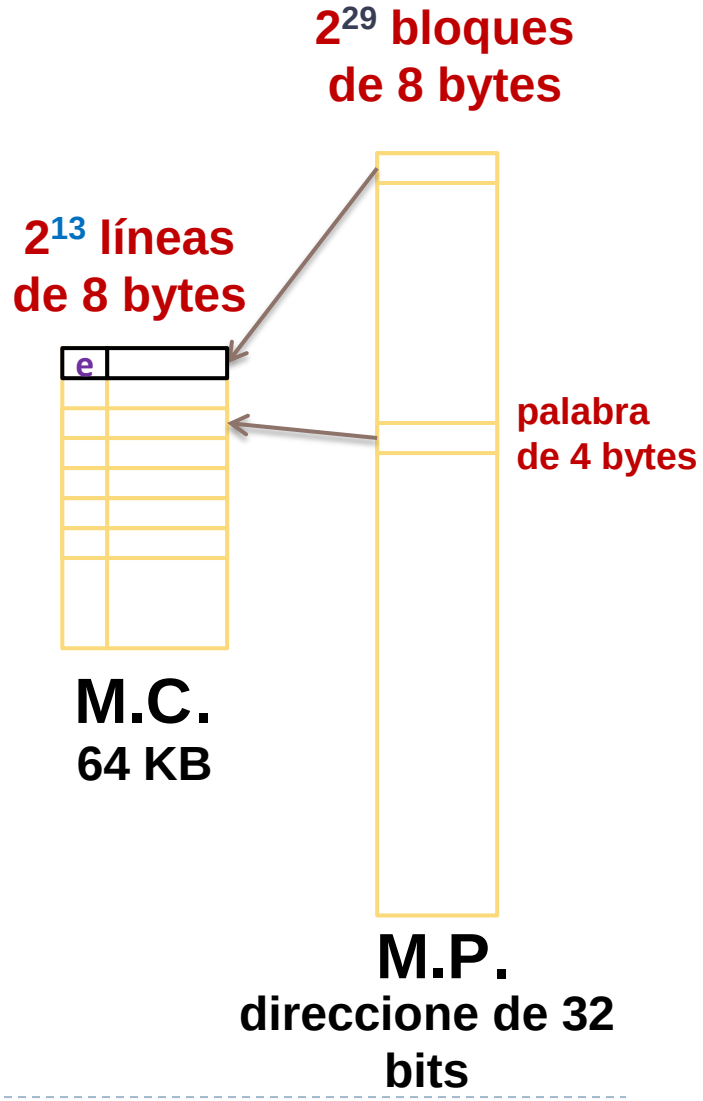


Función de correspondencia asociativa Ejemplo

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

32-3	3
etiqueta	byte

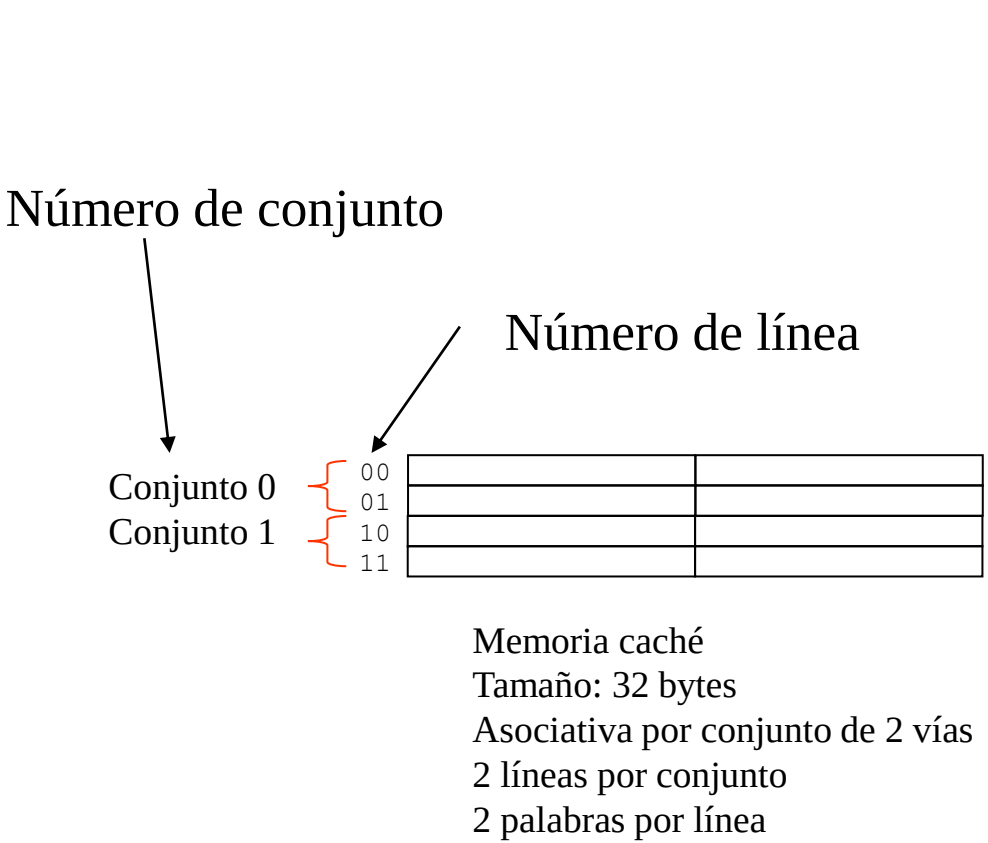
- Si hay una línea con '*etiqueta*' en la caché, está allí el bloque
- Independiente del patrón de acceso, búsqueda costosa
- Etiquetas más grandes: cachés más grandes



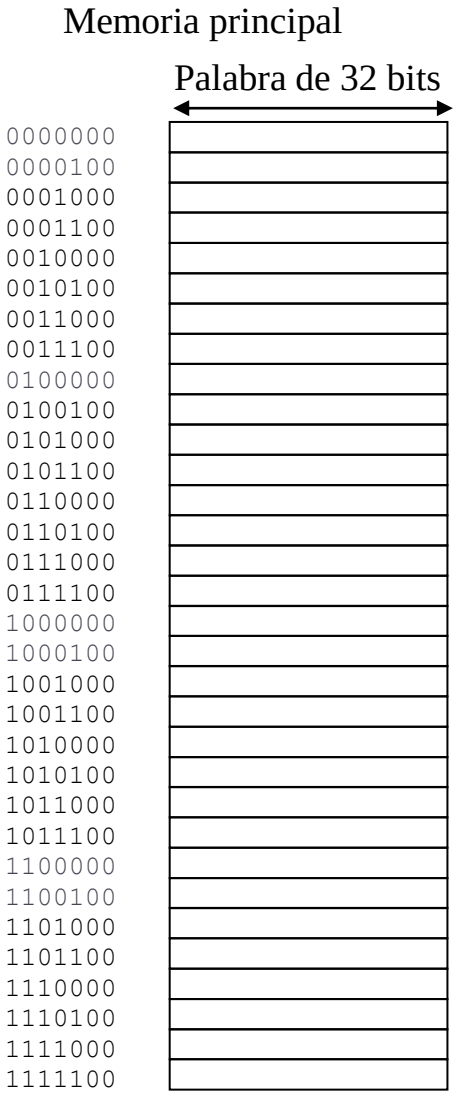
Correspondencia asociativa por conjuntos

- 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 **mod** número de conjuntos
- Dentro de un conjunto el bloque se puede almacenar en cualquiera de las líneas de ese conjunto

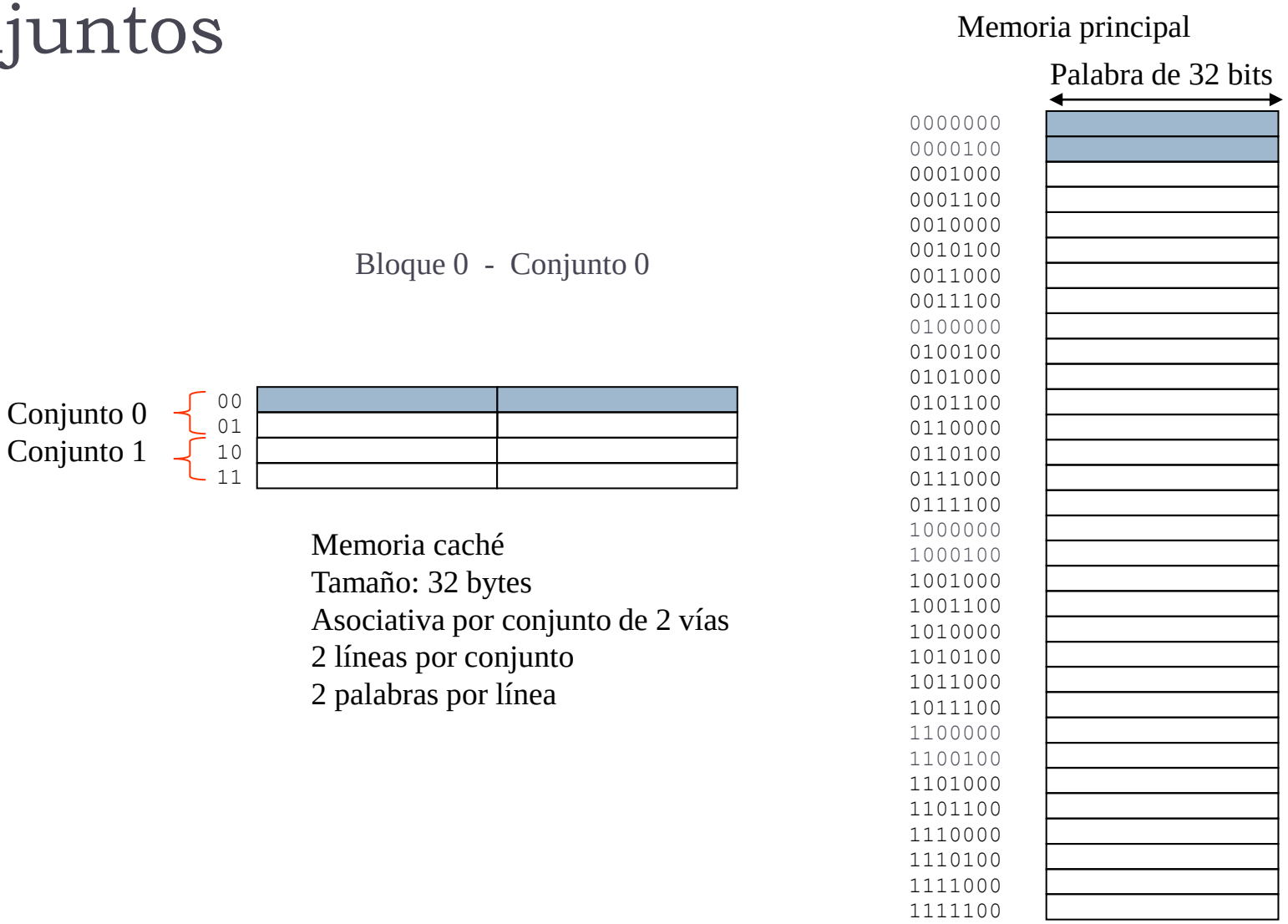
Correspondencia asociativa por conjuntos



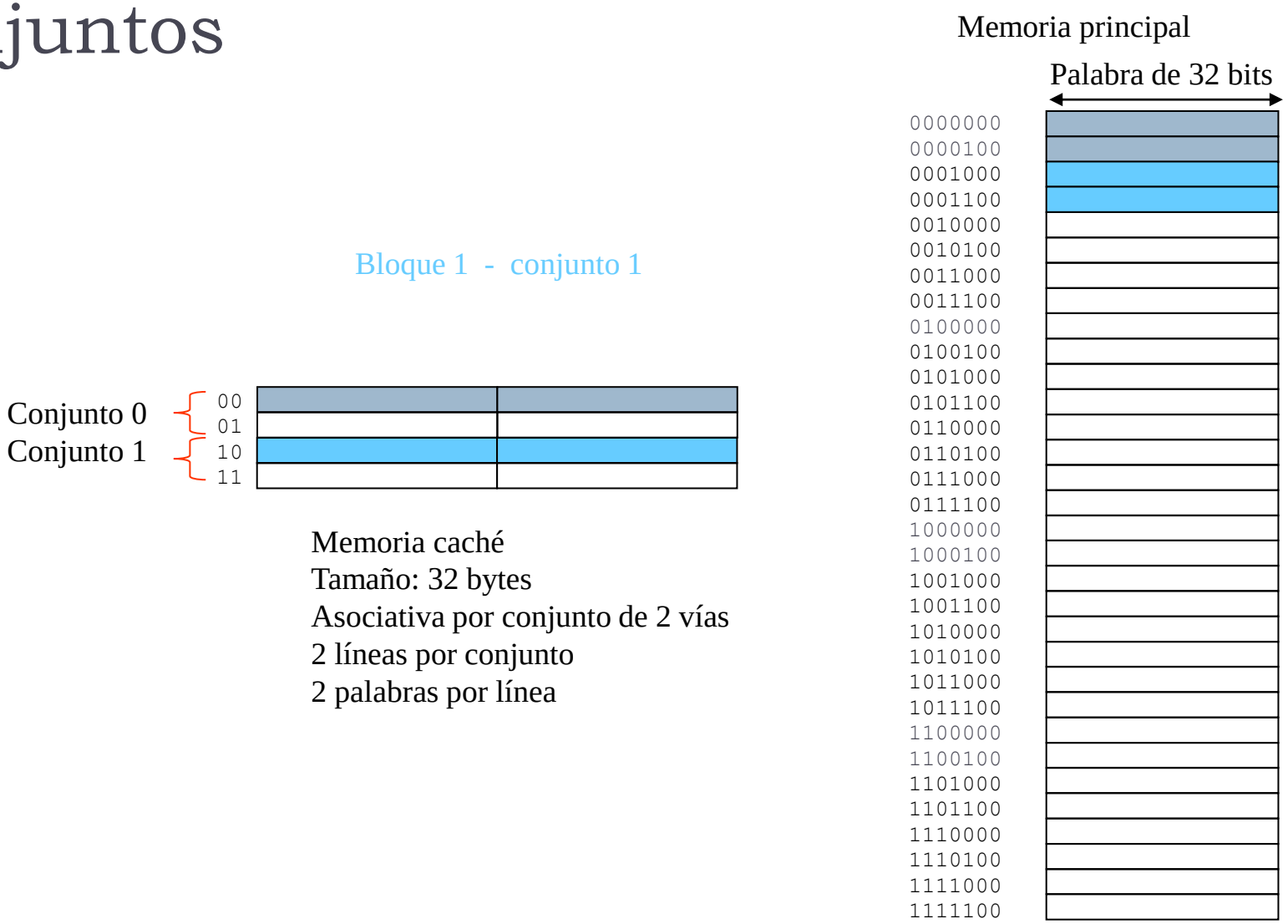
Memoria caché
Tamaño: 32 bytes
Asociativa por conjunto de 2 vías
2 líneas por conjunto
2 palabras por línea



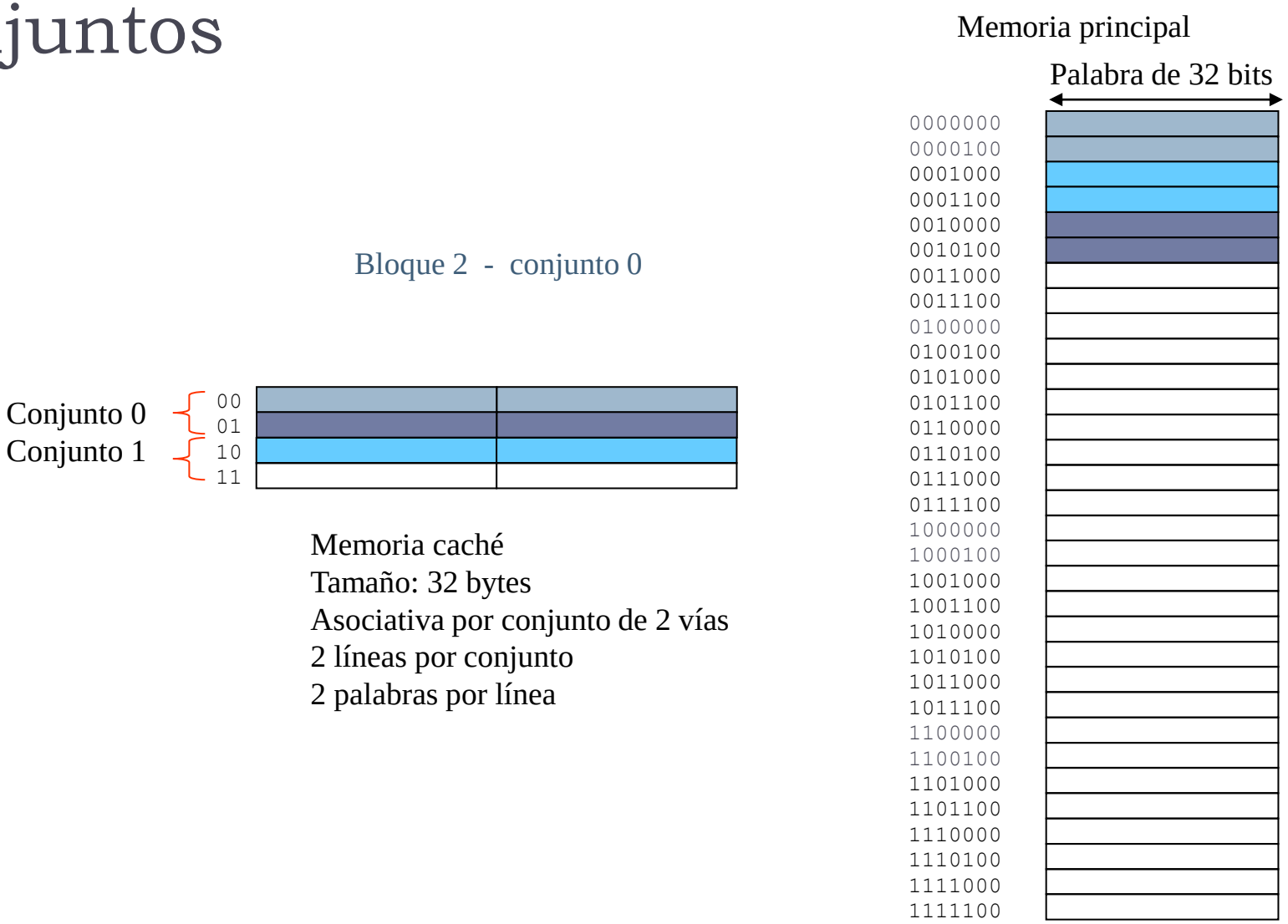
Correspondencia asociativa por conjuntos



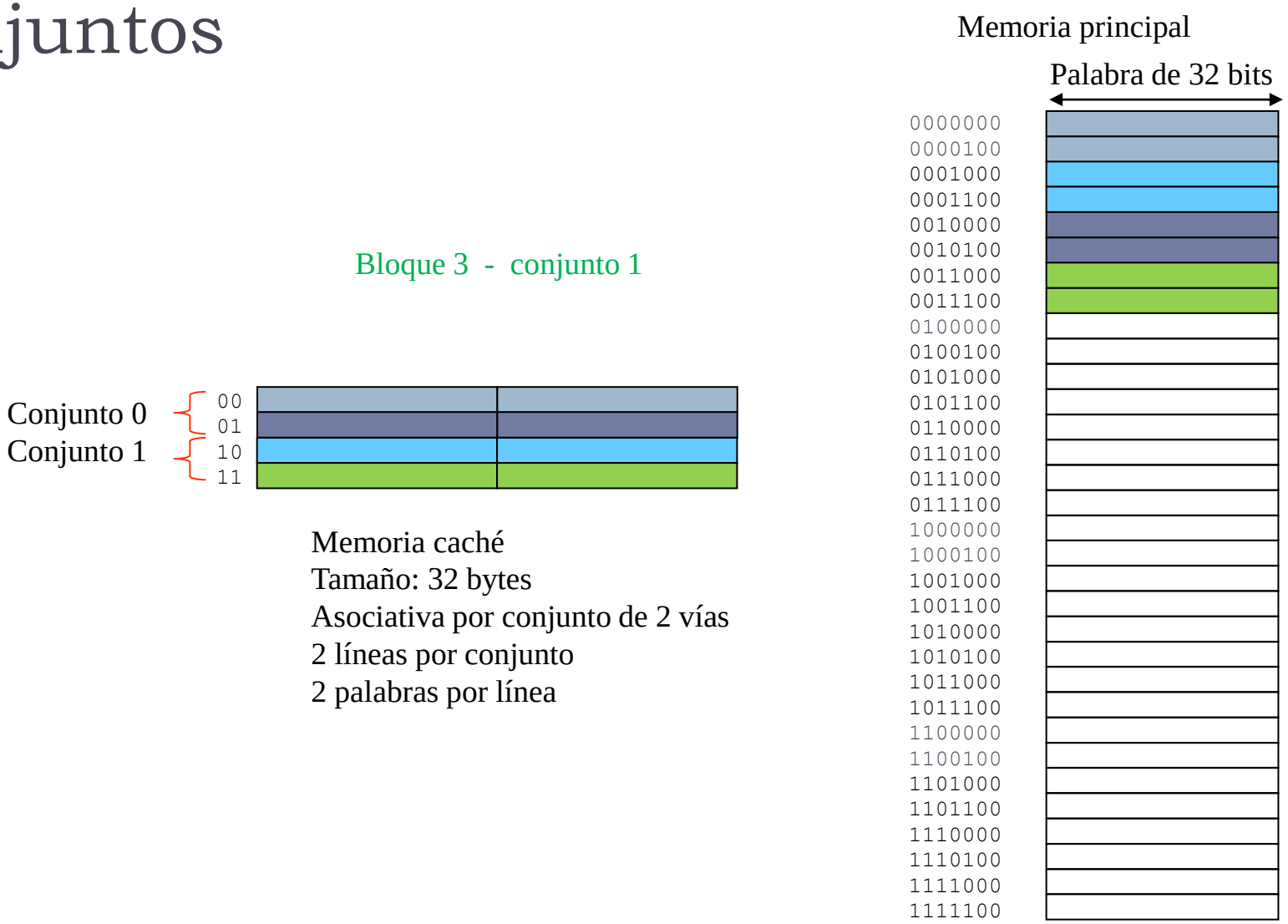
Correspondencia asociativa por conjuntos



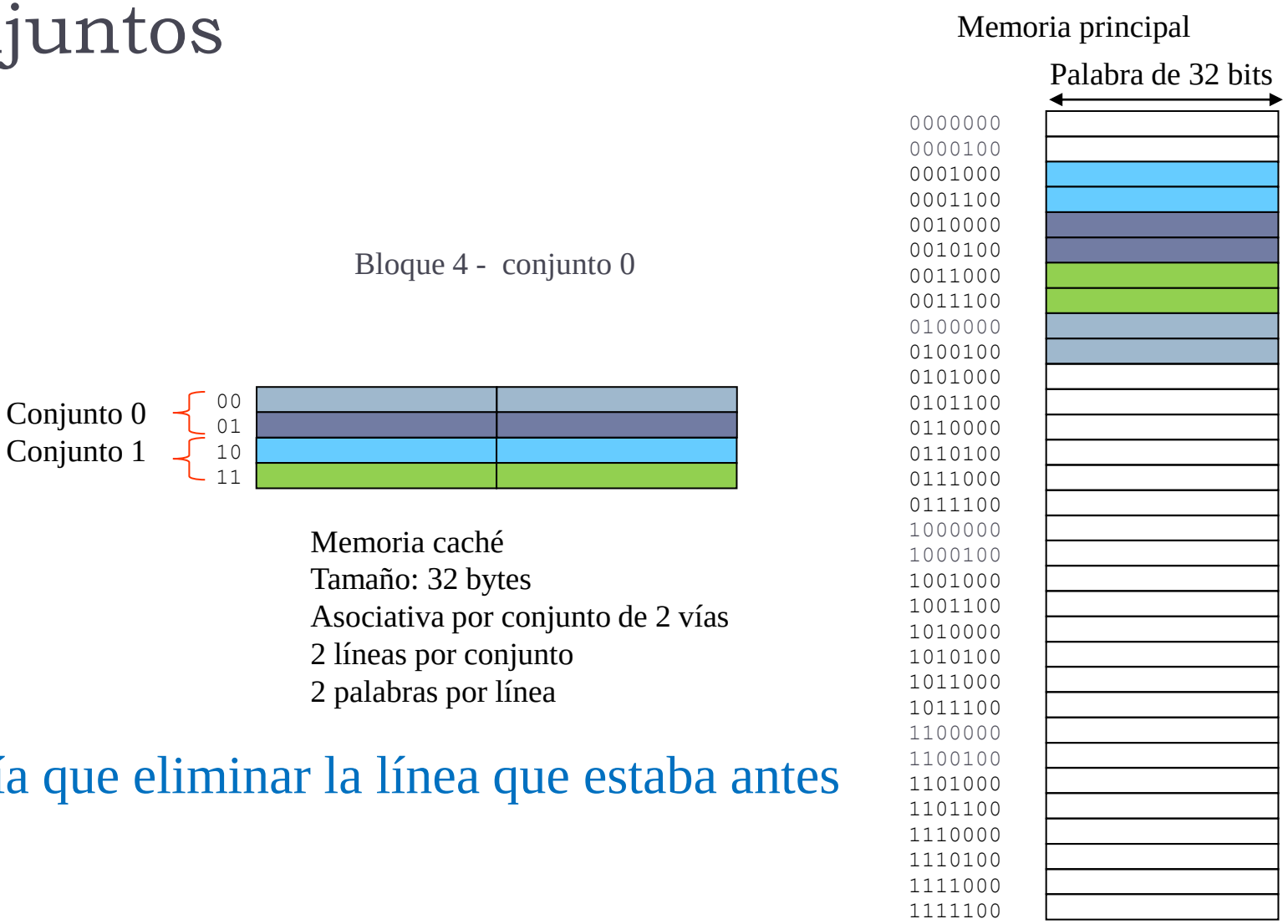
Correspondencia asociativa por conjuntos



Correspondencia asociativa por conjuntos



Correspondencia asociativa por conjuntos

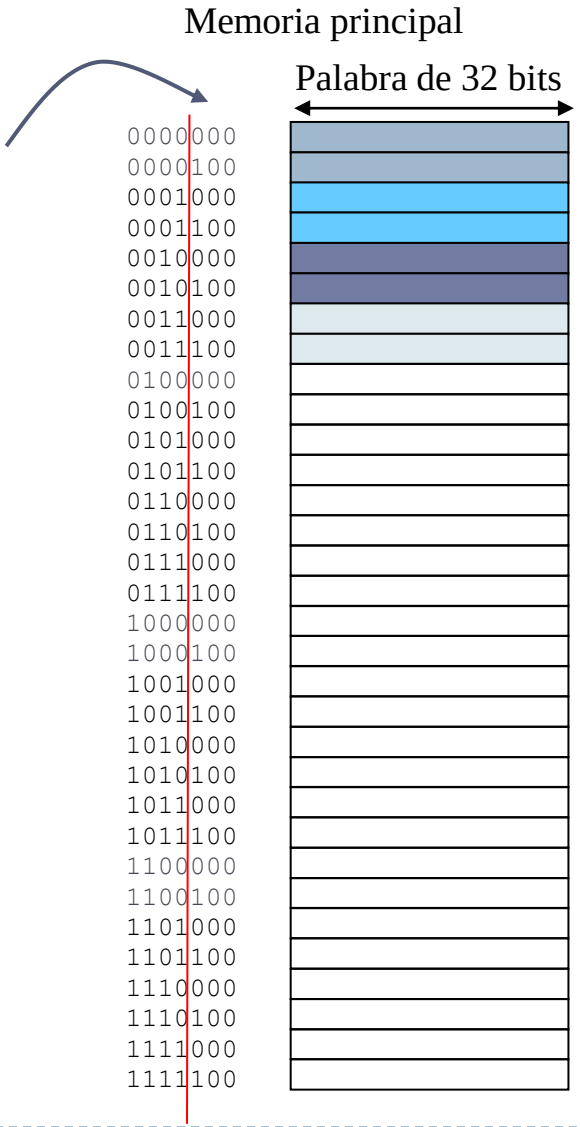


Habría que eliminar la línea que estaba antes

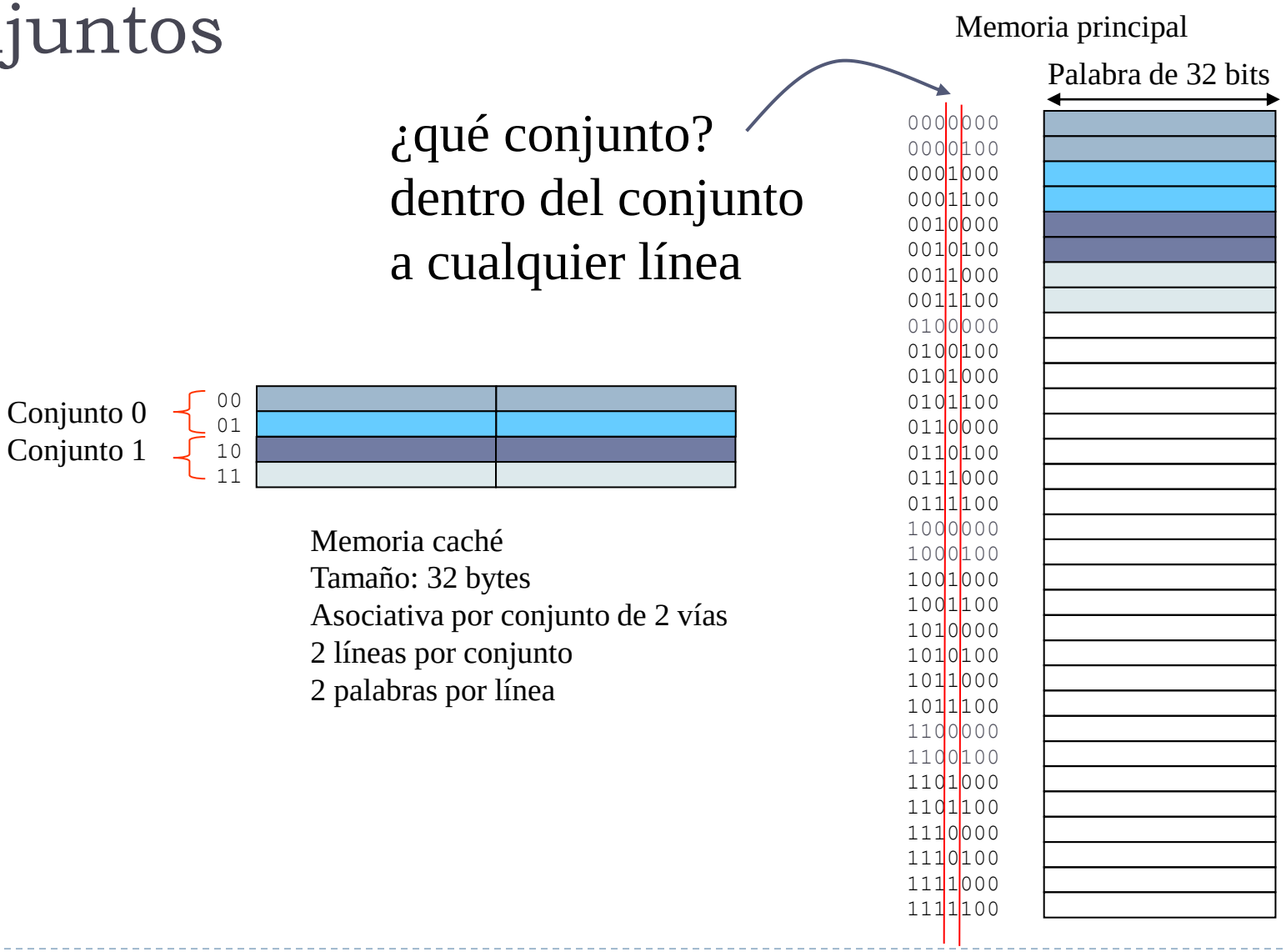
Correspondencia asociativa por conjuntos

¿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

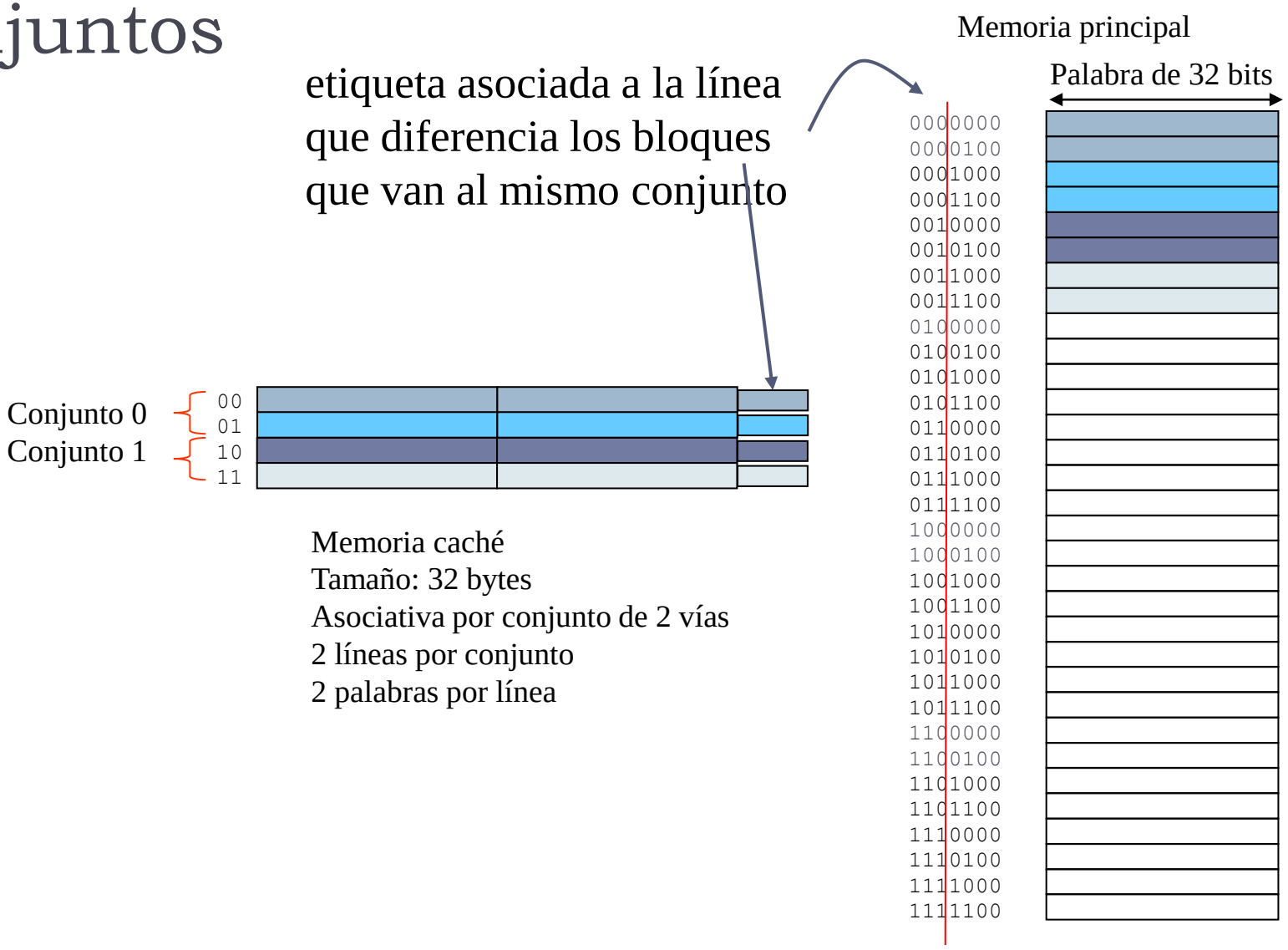


Correspondencia asociativa por conjuntos



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

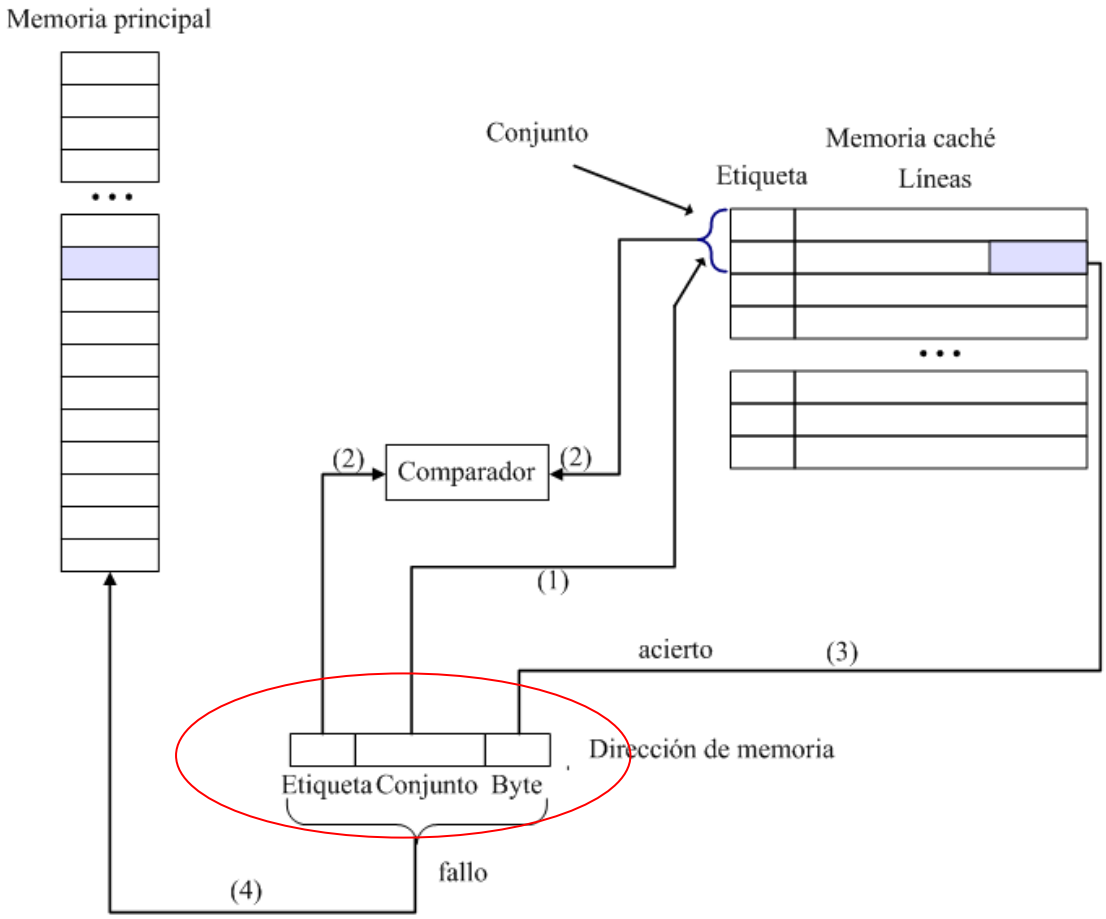
Correspondencia asociativa por conjuntos



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

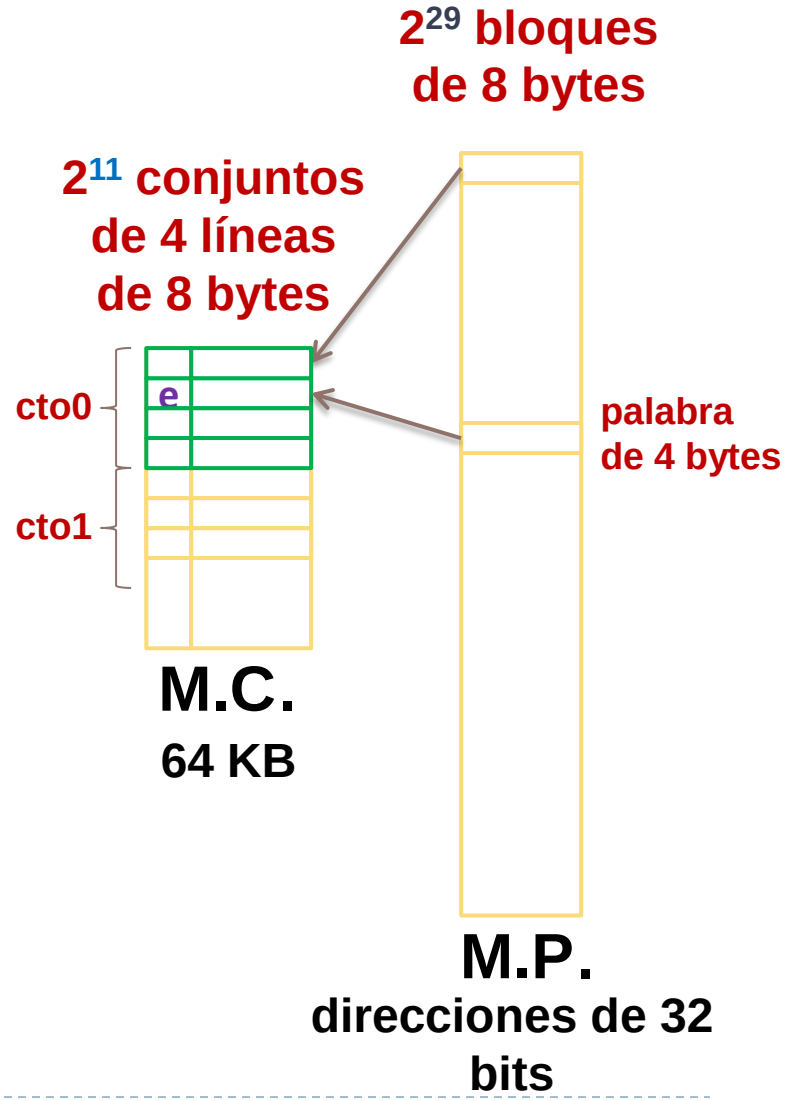


Función de correspondencia asociativa por conjuntos. Ejemplo

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

18	11	3
etiqueta	conjunto	palabra

  - 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.

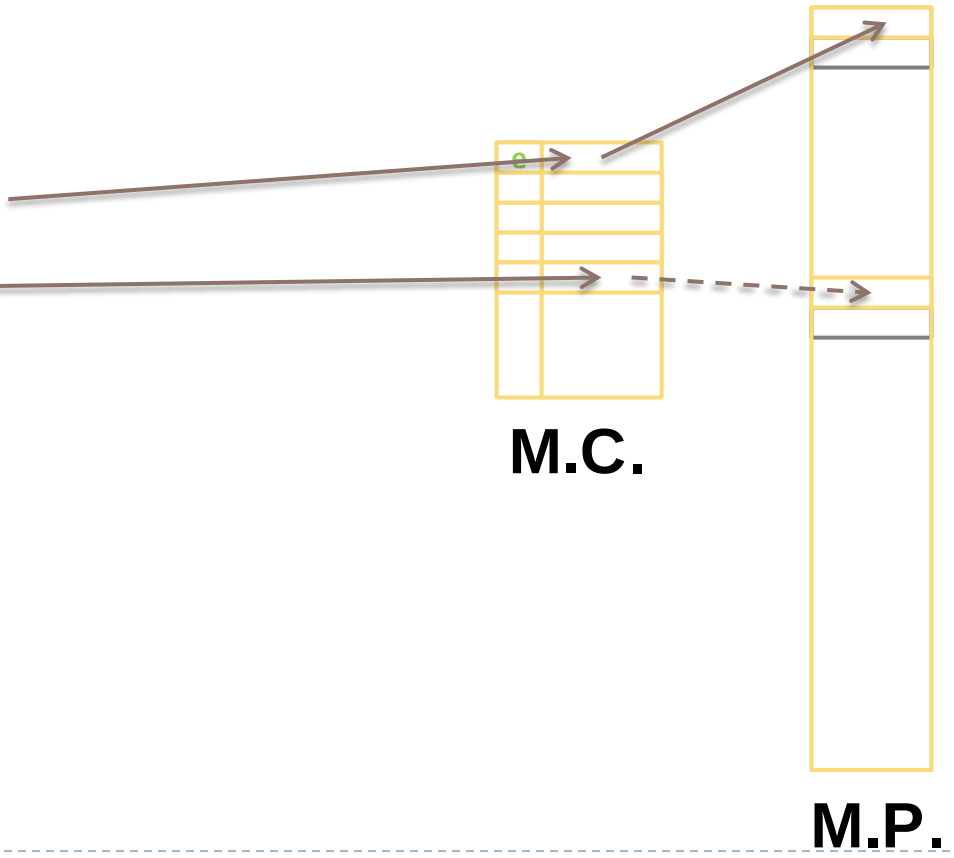
Algoritmos de sustitución

- 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

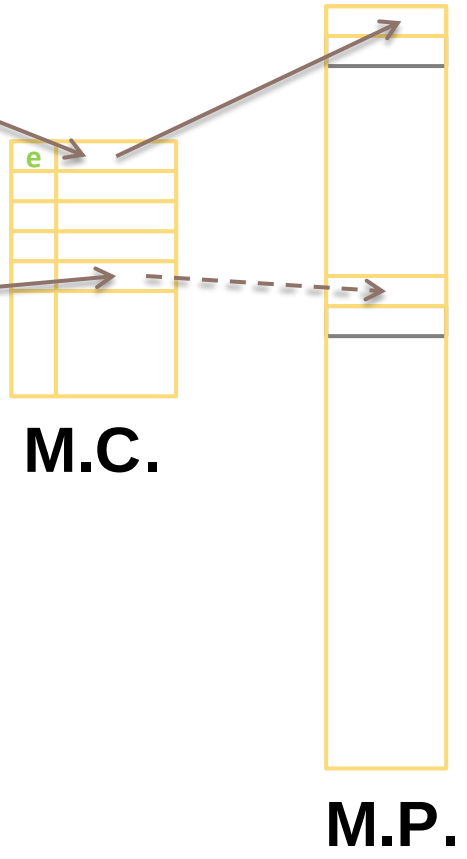
- Cuando se modifica un dato en memoria caché, hay que actualizar en algún momento la memoria principal

- Alternativas:
  - Escritura inmediata.
  - Post-escritura.



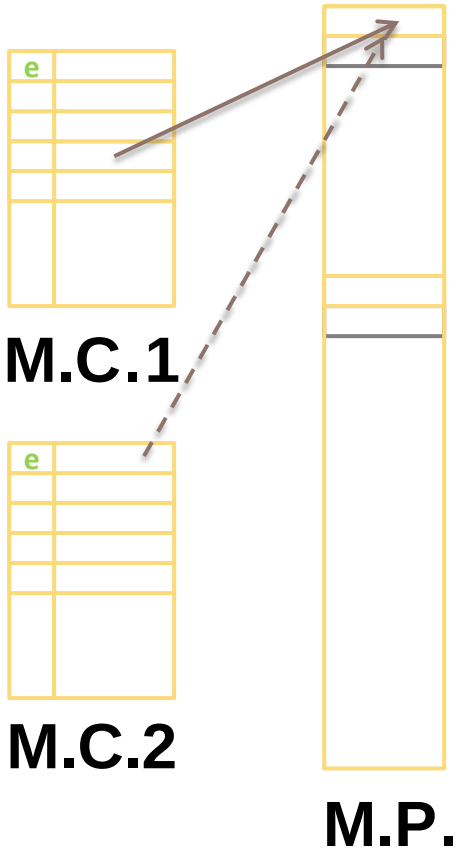
Política de escritura

- 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

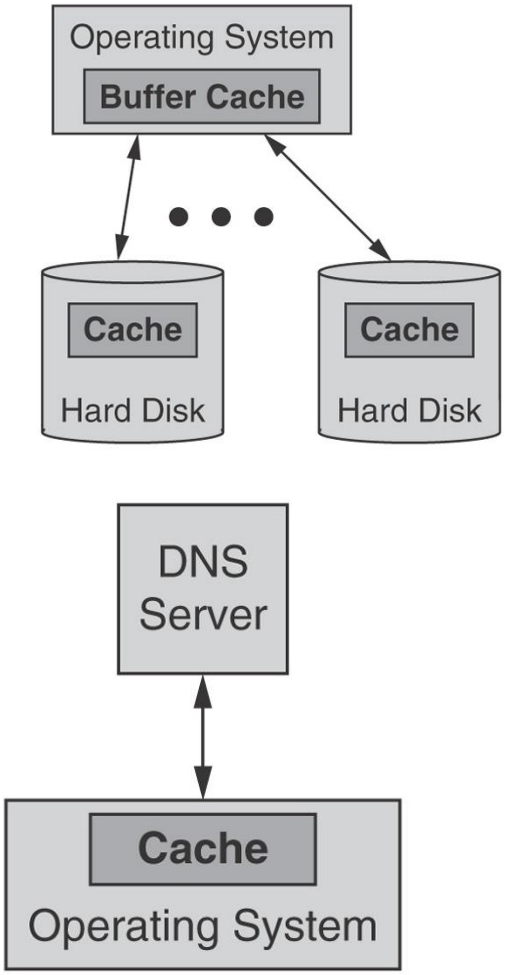


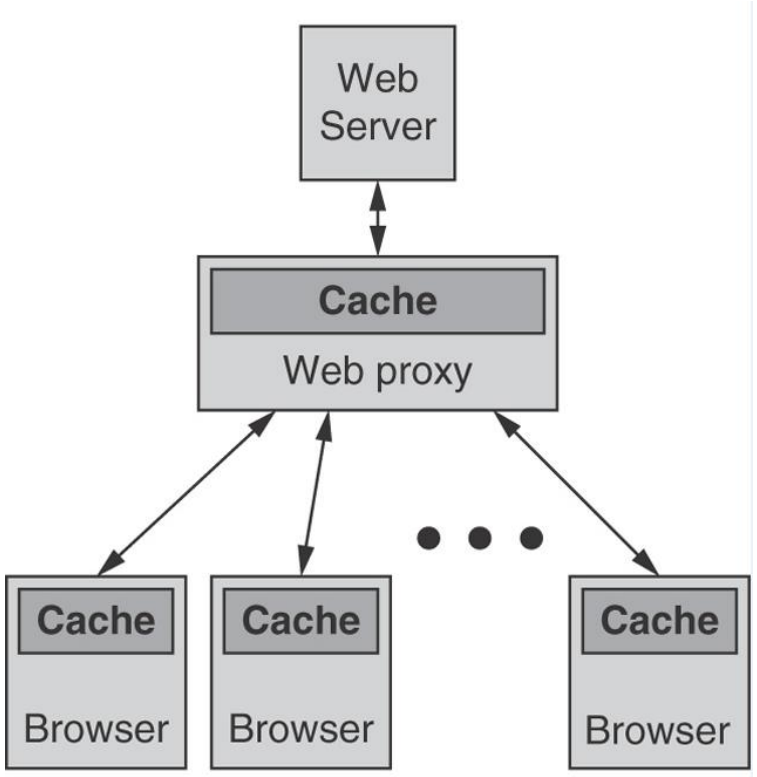
Política de escritura

- 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, ¿cúal 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

Grupo ARCOS

uc3m | Universidad Carlos III de Madrid

Tema 5 (II)
Jerarquía de Memoria

Estructura de Computadores
Grado en Ingeniería Informática

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 almacenaría?
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

nbloques = tamaño memoria / tamaño de línea =
256 MB / 64 =
256 x 2^{20} / 64 = 256 * 2^{14} bloques

b) El número de líneas de memoria caché es

nlineas = tamaño memoria / tamaño de línea =
64 KB / 64 bytes =
2^{16} / 2^6 = 2^{10} = 1024 líneas

Conjuntos = Nº líneas / Nº vías = 1024 / 4 = 256 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.

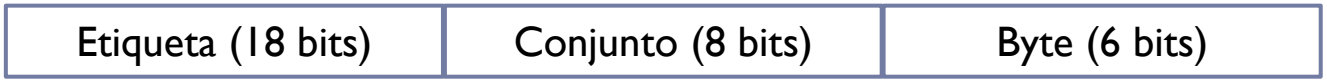
- Byte: el tamaño de la línea es 64 bytes = 2^6bytes.
  Se necesitan, por tanto 6 bits para identificar el byte dentro de la línea.
- Conjunto: Hay 256 conjuntos = 2^8,
  por lo que se necesitan 8 bits para identificar un conjunto
- Etiqueta: para la etiqueta se emplean el resto de los bits de la dirección =
  32 – 6 – 8 = 18

La dirección quedaría:

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

Ejercicio (solución)

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



El byte asociado a esta dirección se encontraría en el conjunto 3

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

- $T_{medio} = tc + (1-h) * T_{fallo}$
- $8 = 5 + (1-h) * 100$

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 esta inicialmente vacía) .