Fundamentos de Computadores

2º Cuatrimestre

2012-2013

MÓDULO 11: Introducción a la jerarquía de memoria

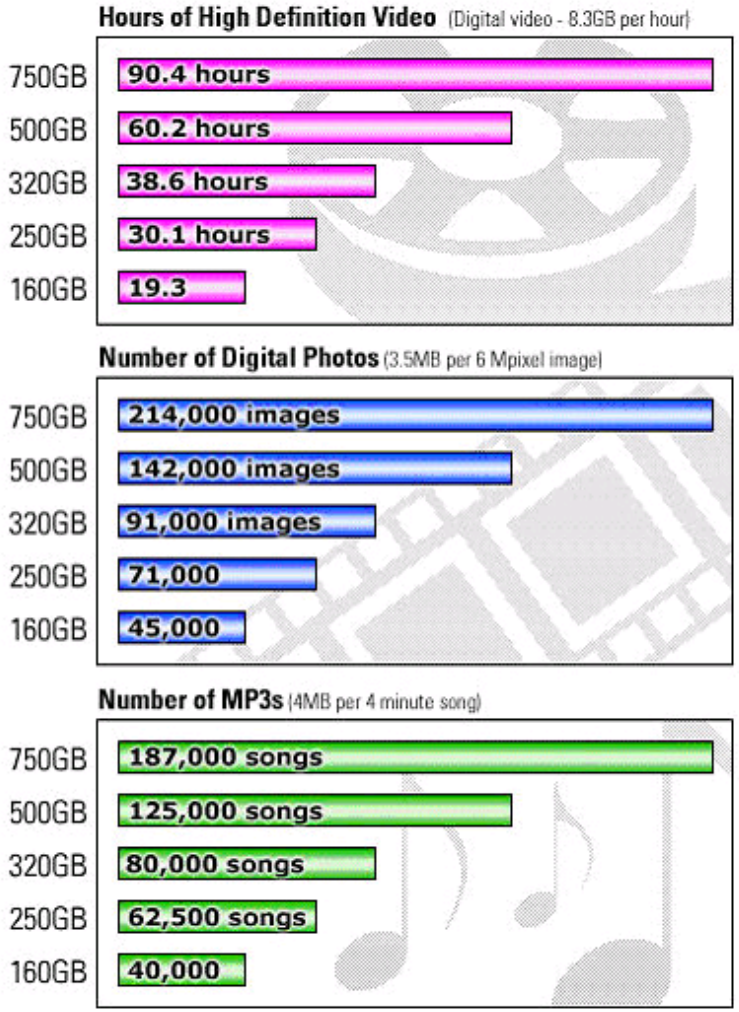
fc²

Sistema de memoria de un computador

- ¿Qué es una memoria? ¿Cómo se implementa?
- ¿Cómo se soluciona el problema del tiempo de acceso a la memoria?
- ¿Por qué TODOS los computadores tienen memoria cache?
- ¿Cómo funciona?

Introducción

- Los datos ocupan espacio en memoria
- Acceder a ellos implica un cierto tiempo



Introducción

- Valores típicos de tiempos de acceso y precios por Gbyte (año 2008)

Tipo de memoria	Tiempo de acceso (ns)	$ por Gbyte
SRAM	0.5-2.5	2000-5000
DRAM	50-70	20-75
Disco magnético	5.000.000-20.000.000	0.20-2

Tipos de memoria semiconductora

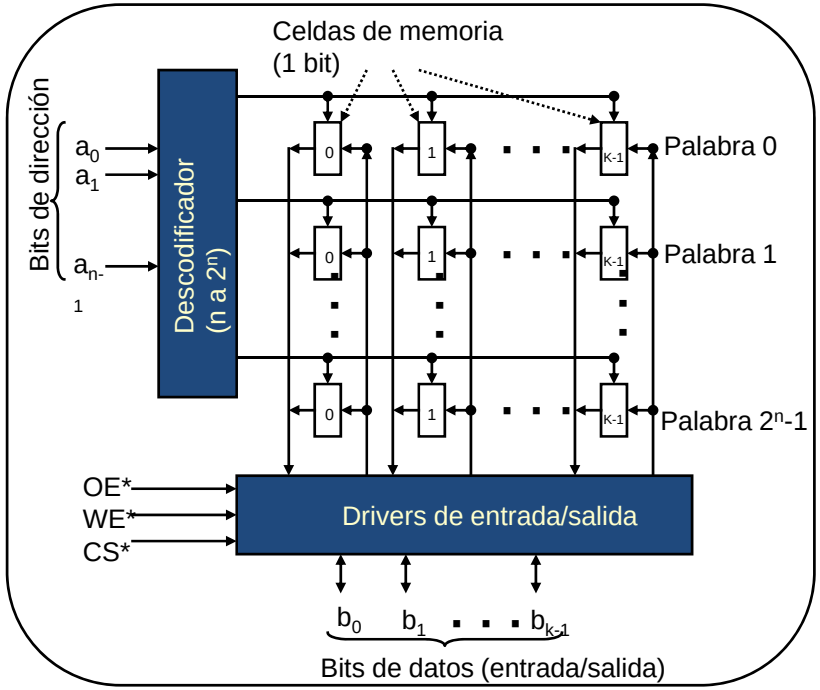
- ¿Cómo está diseñada una memoria por dentro?
  - Sólo vamos a verlo a nivel de módulos, no a nivel de celda básica

Tipos de memoria semiconductora

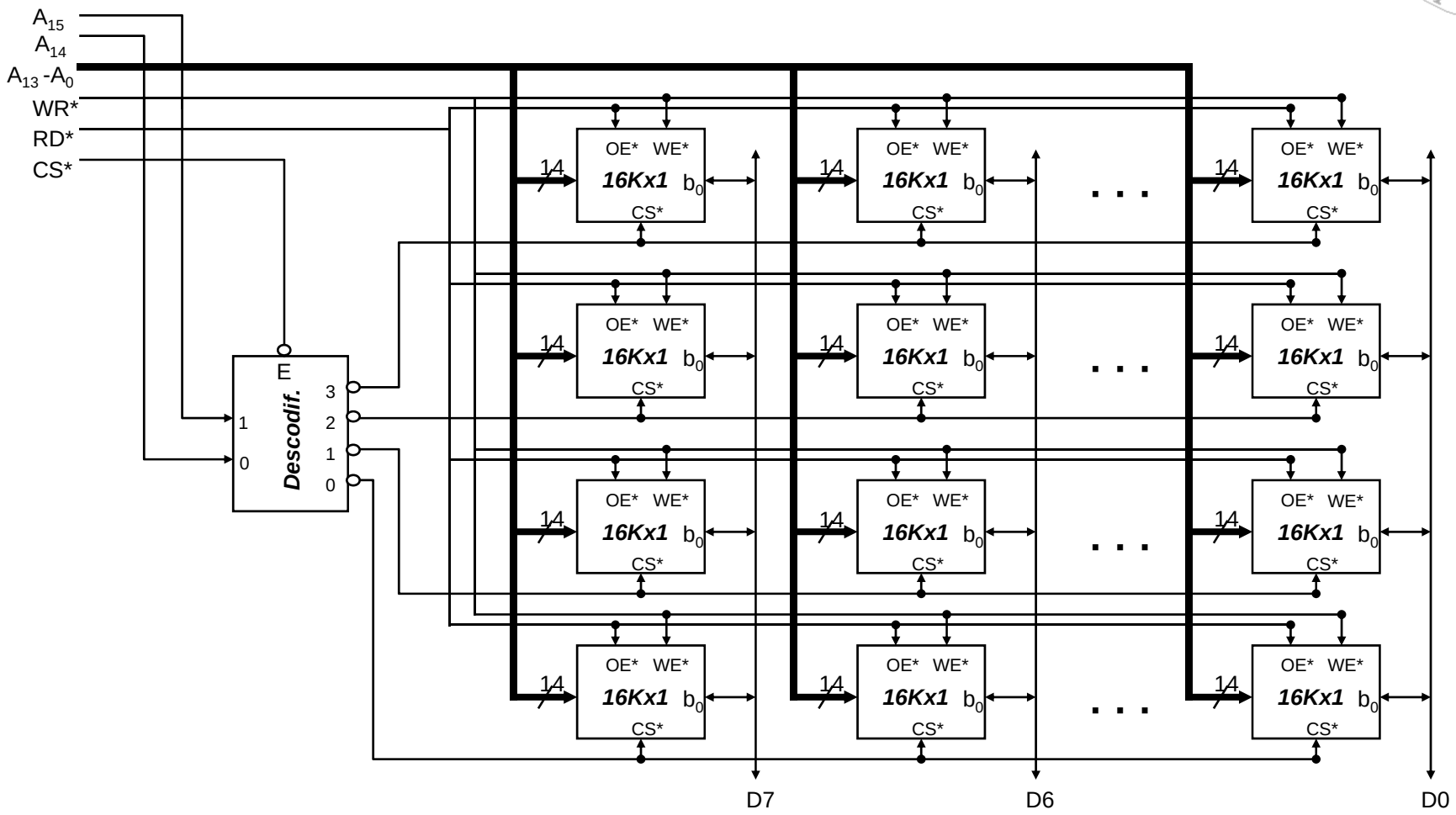
- Los módulos típicos de memoria ROM (PROM, EPROM, etc.) y RAM (SRAM, DRAM) son asíncronos
  - La operaciones de lectura y escritura no se rigen por una señal de reloj
  - Para iniciar una transferencia se utilizan las señales de
    - Dirección
    - Chip select (CS*)
    - Tipo de operación: OE* = Output Enabled (operación de lectura) y WE* = Write Enabled (operación de escritura)
  - Los ciclos de lectura/escritura tienen una duración determinada y conocida para un módulo de memoria específico
- El módulo de memoria debe conectarse al bus del sistema a través de un controlador o adaptador de memoria
  - Convierte las señales de control que genera la CPU a las señales que utiliza el módulo de memoria
  - Genera las señales de control necesarias sobre la CPU para que las transferencias se realice correctamente, conocido el tiempo de ciclo de la memoria

Memoria RAM estática (SRAM)

- Celda básica (1 bit) constituida por un biestable
  - Mantiene la información mientras exista suministro eléctrico
  - La implementación puede ser bipolar o MOS
- Ventajas
  - Tiempo de acceso y de ciclo reducido
    - Alta velocidad de transferencia
- Desventajas
  - Disipan mucha energía
  - Baja densidad de integración
  - Coste elevado
- Aplicación
  - Diseño de memorias cache
    - Alta velocidad y pequeño tamaño
- Chips de memoria SRAM comerciales
  - Tamaño 2^nxk
    - 2^n palabras de k bits, siendo k = 1, 2, 4 u 8
  - Ejemplo: chips de memoria 51C86 de Intel
    - Tamaño 4Kx4
  - Organización 2-D
    - Un chip de 2^n palabras de k bits se organiza como un array de 2^n filas por k columnas
  - Para construir placas de memoria de mayor tamaño se deben combinar varios chips de memoria con la lógica de descodificación adicional adecuada

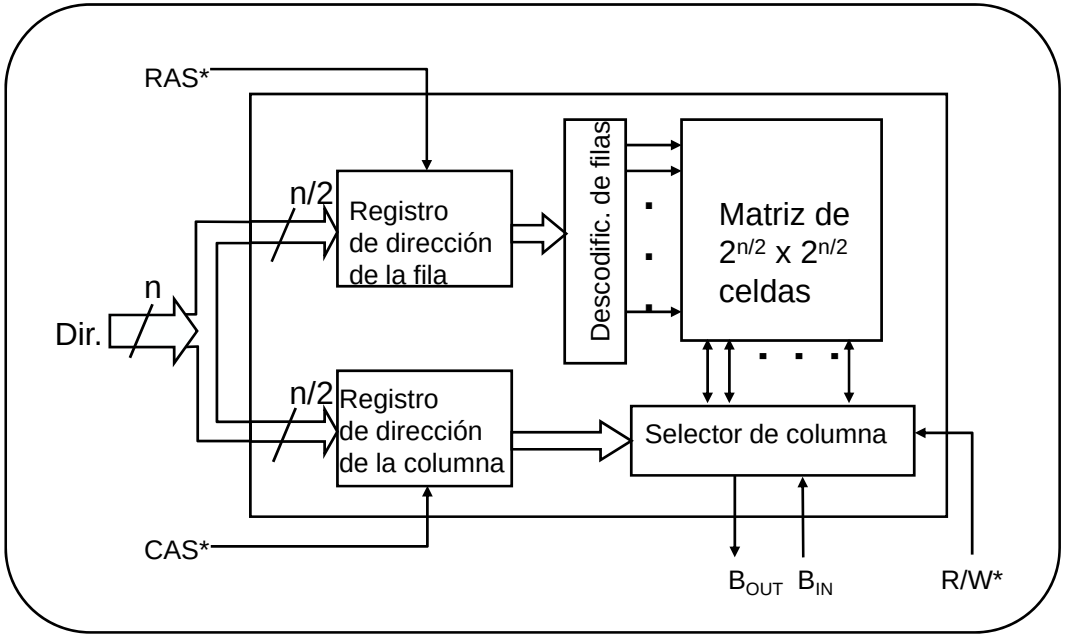


Memoria RAM estática (SRAM)



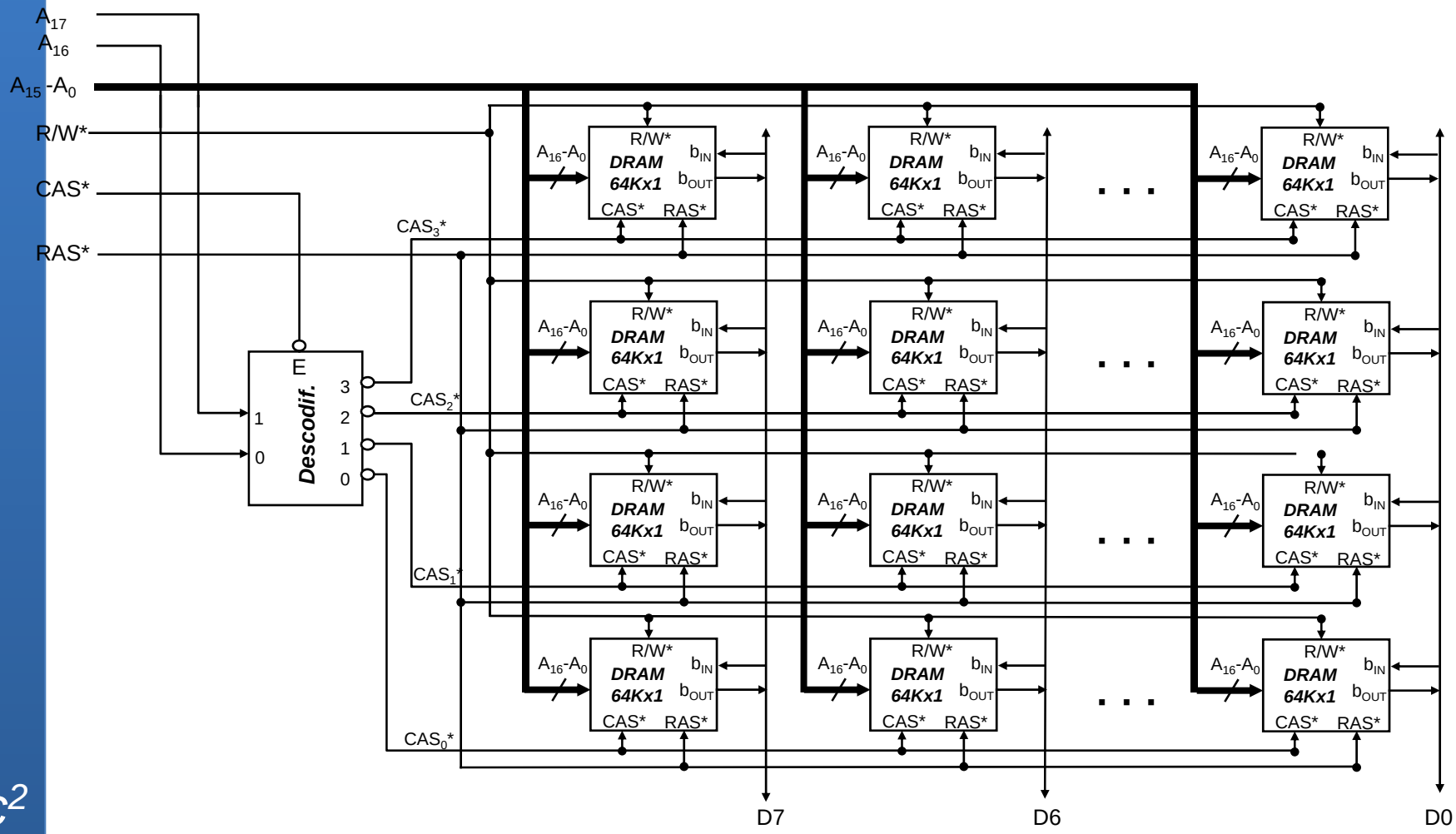
Memoria RAM dinámica (DRAM)

- Celda básica (1 bit) constituida por un condensador y un único transistor MOS
  - La información binaria se mantiene en forma de carga del condensador
    - 1 lógico: presencia de carga en el condensador; 0 lógico: ausencia de carga en el condensador
  - El control de la carga/descarga del condensador se realiza mediante un conmutador
    - Implementado mediante un transistor MOS
  - Los condensadores pierden la carga al cabo de unos pocos milisegundos
    - Necesidad de un refresco periódico de la información contenida en la RAM
    - El refresco consiste en la lectura del condensador y posterior escritura con idéntico valor (amplificado)
- Ventajas
  - Bajo consumo de energía
  - Alta densidad de integración
  - Coste reducido
- Desventajas
  - Tiempo de ciclo elevado
  - Necesidad de refresco
- Aplicación
  - Diseño de memoria principal
    - Gran tamaño y velocidad baja
- Chips de memoria DRAM
  - Tamaño 2^nx1 ó 2^nx4
  - Organización 2-1/2-D
    - Un array $2^{n/2}$x$2^{n/2}$ por bit



RAS*: Línea de selección de la fila
CAS*: Línea de selección de la columna

Memoria RAM dinámica (DRAM)



Jerarquía de Memoria

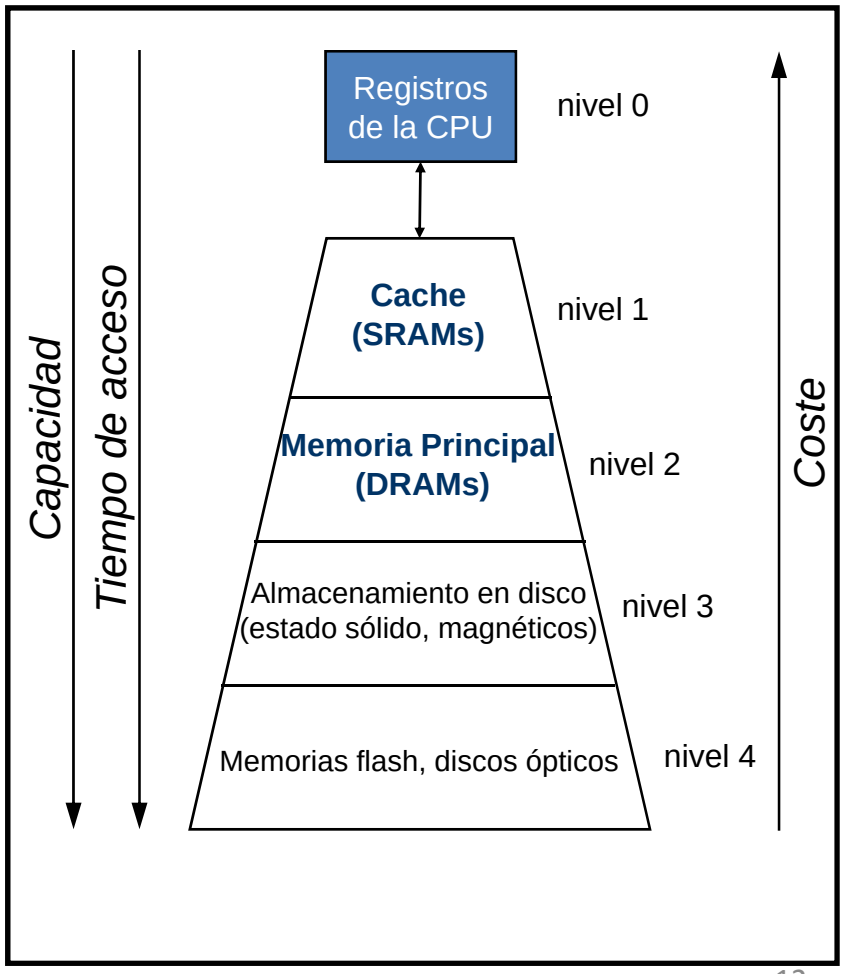
- Objetivo:
  - Conseguir una memoria de gran tamaño, rápida y al menor coste posible.
  - De forma transparente al usuario
- ¿Cómo organizar la memoria?
- Base: Principio de localidad

  "Cualquier programa accede a una porción relativamente pequeña de su espacio de direcciones en cualquier instante de tiempo"

Jerarquía de memoria

Niveles de la Jerarquía de memoria

- Un computador típico está formado por diversos niveles de memoria, *organizados de forma jerárquica*:
  - Registros de la CPU
  - Memoria Cache
  - Memoria Principal
  - Memoria Secundaria (discos)
  - Memorias flash y CD-ROMs
- El coste de todo el sistema de memoria excede al coste de la CPU
  - Es muy importante optimizar su uso



fc²

Principio de localidad

- Localidad temporal: Un dato usado en un determinado instante tiende a ser pronto reutilizado
- Localidad espacial: Si un dato es utilizado en un determinado instante, es muy probable que los datos cercanos a él sean también pronto utilizados

Localidad espacial

- Es habitual que un programa acceda datos que están almacenados en posiciones consecutivas
- Los arrays se almacenan en posiciones consecutivas y se suelen acceder secuencialmente

```
sum= 0;
for (i = 0; i < MAX; i++)
        sum= sum + a[i];
```

Localidad temporal

- Es habitual que un programa acceda la misma variable varias veces
- Lo óptimo es mantenerla en registro, pero no siempre es posible

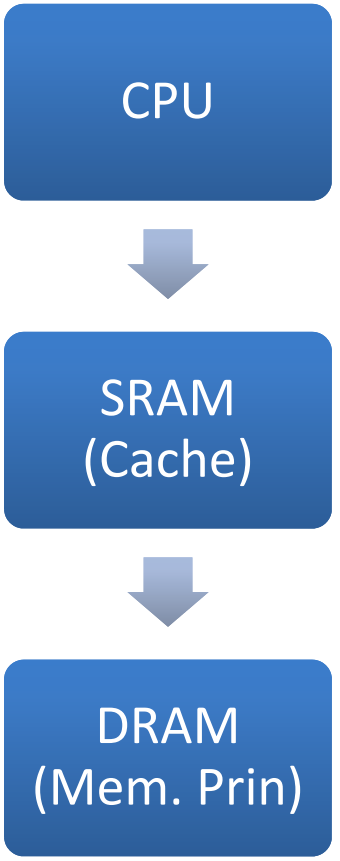
```
sum= 0;
for (i = 0; i < MAX; i++)
        sum= sum + a[i];
```

Jerarquía de memoria

- Jerarquía de memoria: múltiples niveles de memoria de diferentes velocidades y tamaños
- Se basa en el principio de localidad
- Mantiene los datos usados recientemente cerca del procesador (localidad temporal)
- Mueve junto con el dato pedido, otros datos cercanos a él, a niveles de memoria más cercanos al procesador (localidad espacial)

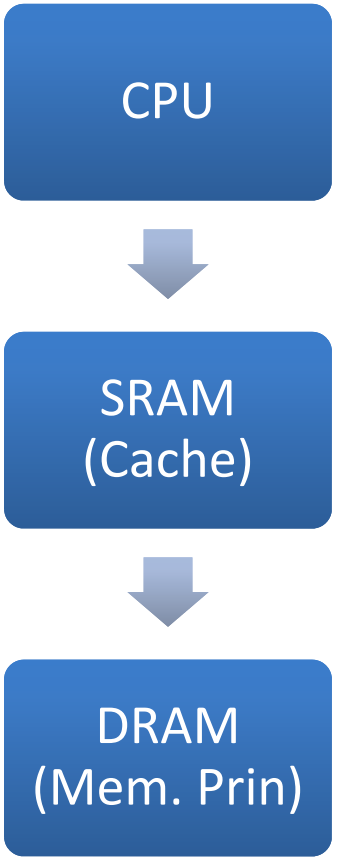
¿Cómo explotar la localidad temporal?

- Cada vez que se accede a memoria principal, llevamos una copia del dato a una memoria más pequeña y rápida (SRAM)
  - Introduce una penalización en el primer acceso
  - Será amortizada si se producen más accesos, pues se usará la copia y se evitará un acceso a memoria principal.



¿Cómo explotar la localidad espacial?

- Cada vez que se accede a la dirección i, llevamos una copia del dato a la cache
- Pero además, se copian varios datos más
  - Por ejemplo, si se accede al elemento a[0], se traen a la cache a[0],a[1],a[2] y a[3]
  - De nuevo, hay una penalización inicial que suele amortizarse.



Jerarquía de memoria

- La memoria más rápida (cache) se coloca más cerca del procesador ➡ Accesos más rápidos
- Y el nivel de memoria más lento será el más alejado
- La búsqueda de los datos comienza siempre en la cache, y de no encontrarse en ella, continúa por los demás niveles de memoria

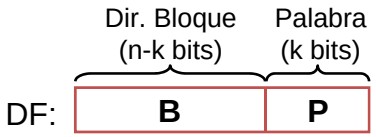
Ilusión: memoria rápida

Gestión de la jerarquía de memoria

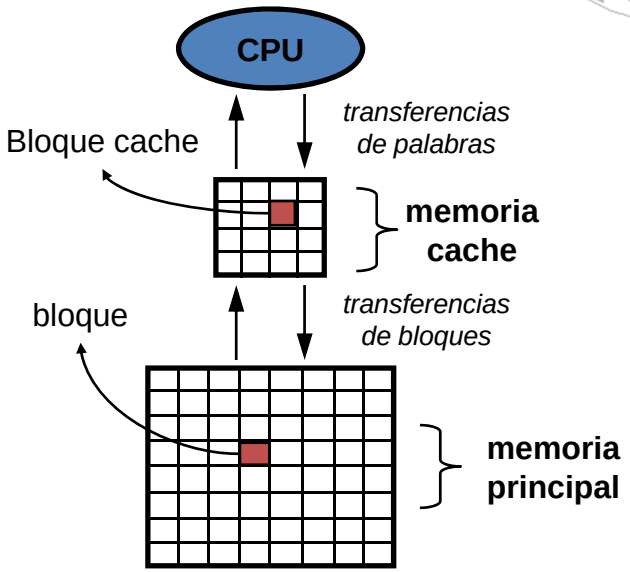
- **Vamos a trabajar con una memoria en 2 niveles**
  - Cache y memoria principal
- **Bloque**: unidad mínima de transferencia entre los dos niveles
- **Acierto (hit):** el dato solicitado está en la cache
- **Fallo (miss):** el dato solicitado no está en la cache y es necesario buscarlo en la memoria principal

Memoria Cache

- **Memoria pequeña y rápida** situada entre el procesador y la memoria principal
  - Almacena una copia de la porción de información actualmente en uso de la memoria
- **Objetivo**: disminuir el tiempo de acceso a memoria
- **Estructura** del sistema memoria cache/principal:
  - *MP (memoria principal):*
    - formada por 2^n palabras direccionables
    - "dividida" en nB bloques de tamaño fijo de 2^K palabras por bloque
    - Campos de una dirección física:



  - *MC (memoria cache):*
    - formada por nM bloques (o líneas) de 2^K palabras cada uno (nM<<nB)
  - *Directorio (en memoria cache):*
    - Para cada bloque de MC, indica cuál es el bloque de MP que está alojado en él.

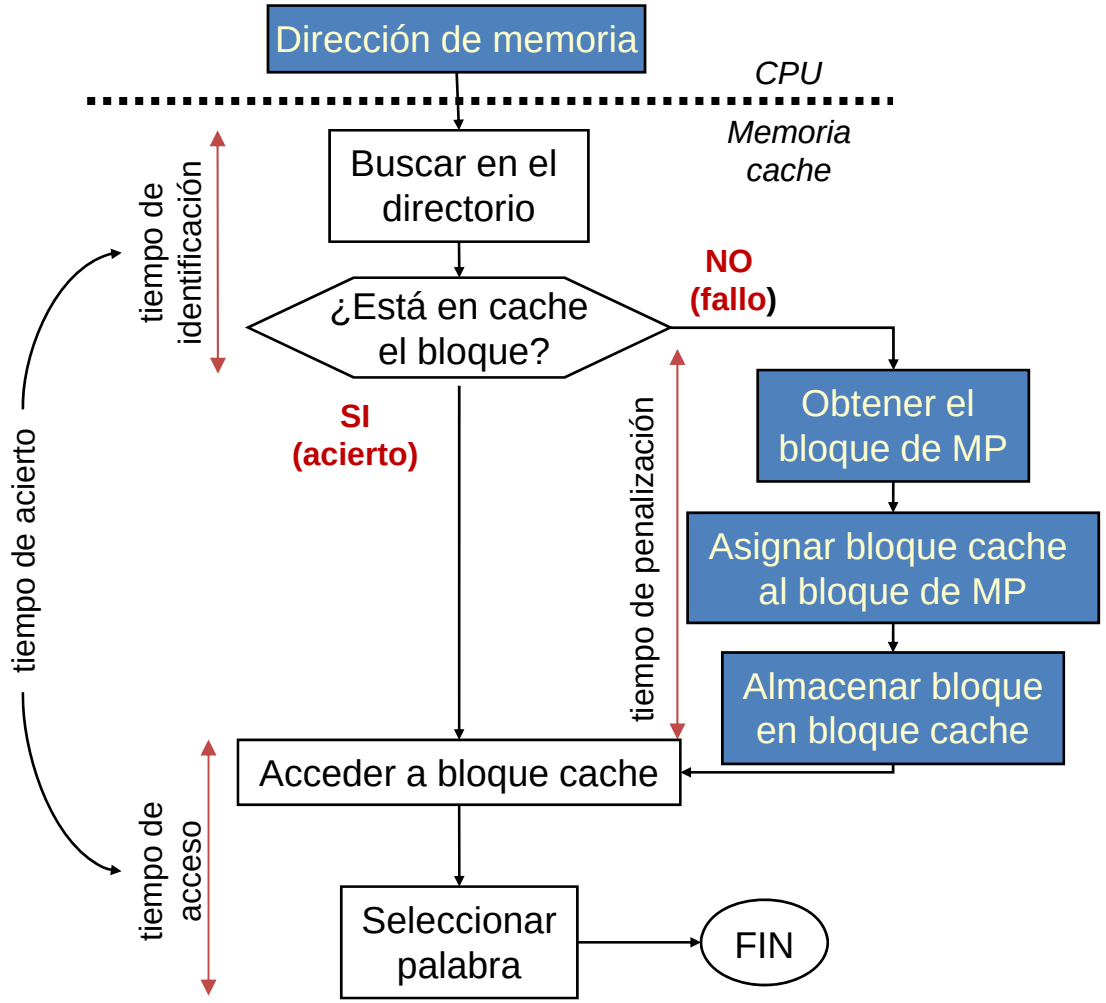


nB: número de bloques
nM: número de bloques de cache
B: dirección del bloque
M: dirección del bloque de cache
P: palabra dentro del bloque

Gestión de la jerarquía de memoria

- Cuando la CPU genera una referencia, busca en la cache
  - Si la referencia no se encuentra en la cache: **FALLO**
  - Cuando se produce un fallo, no solo se transfiere una palabra, sino que se lleva un **BLOQUE** completo de información de la MP a la MC
  - *Por la propiedad de localidad temporal*
    - Es probable que en una próxima referencia se direccione la misma posición de memoria
    - Esta segunda referencia no producirá fallo: producirá un **ACIERTO**
  - *Por las propiedades de localidad espacial*
    - Es probable que próximas referencias sean direcciones que pertenecen al mismo bloque
    - Estas referencias no producen fallo

Memoria cache



Principales objetivos:

- Maximizar la tasa de aciertos
- Minimizar el tiempo de acceso
- Minimizar el tiempo de penalización
- Reducir el coste hardware

Tiempo medio de acceso a M (T_{MAM})

$$T_{MAM} = T_{acierto} + (1 - H)T_{penalizacion}$$

(frecuencia de hits)

Memoria Cache

- ¿Cómo sabemos que un dato está en la cache?
- Y si está, ¿cómo lo encontramos?

POLÍTICAS DE EMPLAZAMIENTO

Políticas de emplazamiento

- Política de emplazamiento:
  - Necesaria ya que existen menos bloques en MC que bloques en MP
  - Determina en qué bloque, o bloques, de MC, puede cargarse cada bloque de MP
- Existen diferentes políticas:
  - **Emplazamiento directo**
  - Emplazamiento asociativo
  - Emplazamiento asociativo por conjuntos

Políticas de emplazamiento

Emplazamiento directo:

Cada inquilino tiene asignado un **único** piso donde alojarse

Cada bloque puede ir a un único lugar de la cache

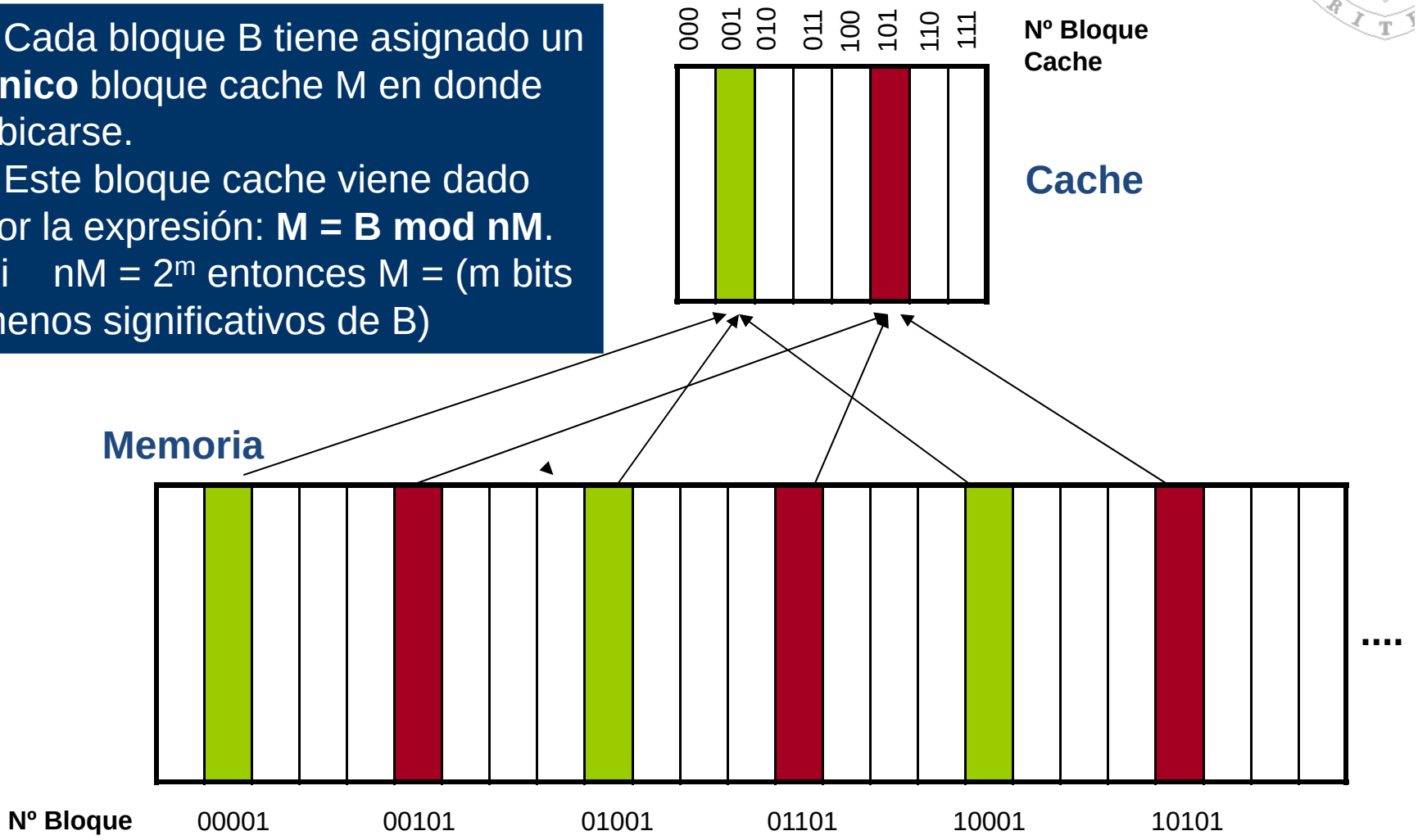
→ **Búsqueda rápida**

↓

La localización en la cache se basa en la dirección del bloque en memoria

Emplazamiento directo

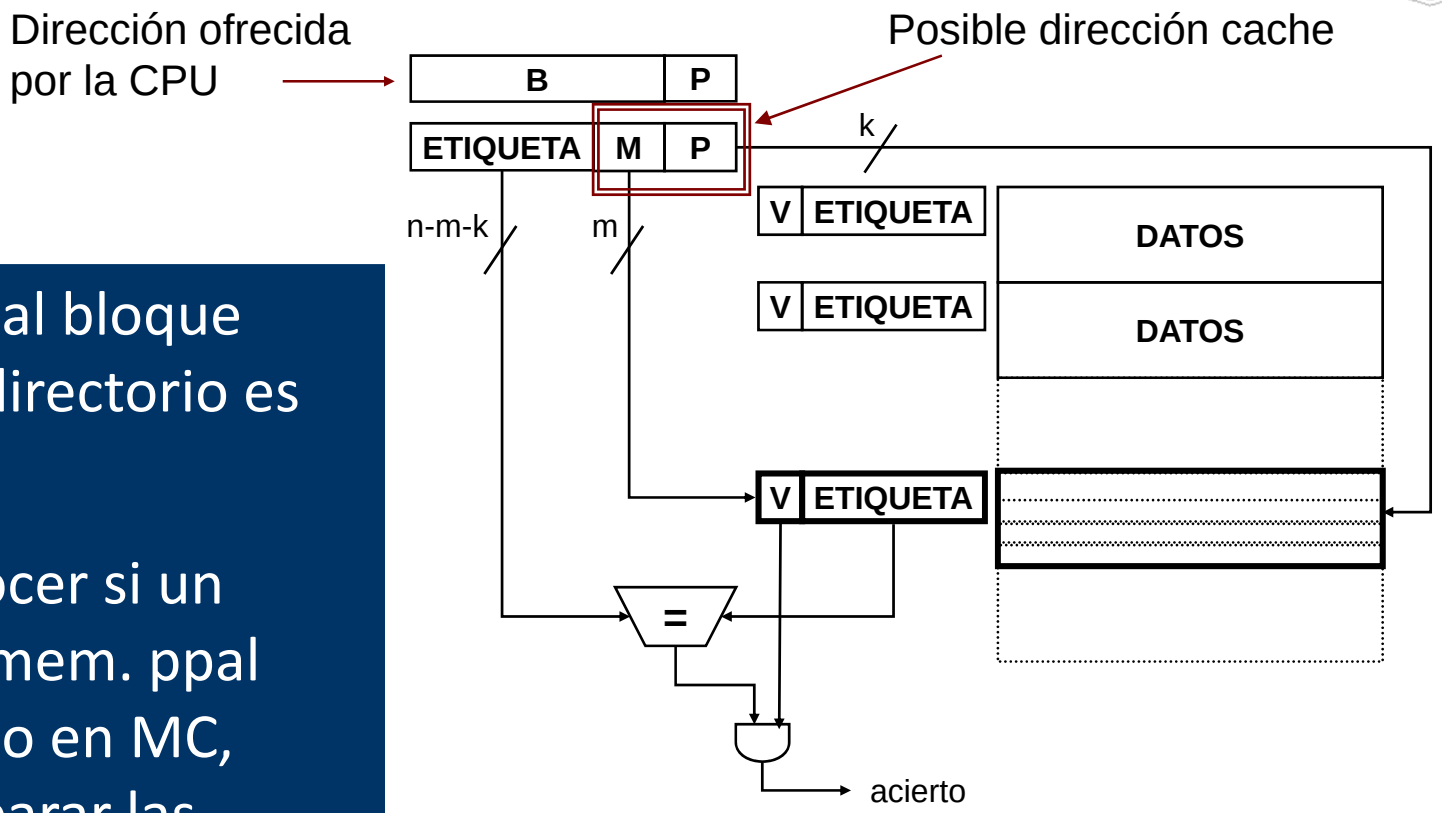
- Cada bloque B tiene asignado un **único** bloque cache M en donde ubicarse.
- Este bloque cache viene dado por la expresión: **M = B mod nM**. Si nM = 2^m entonces M = (m bits menos significativos de B)



Emplazamiento directo

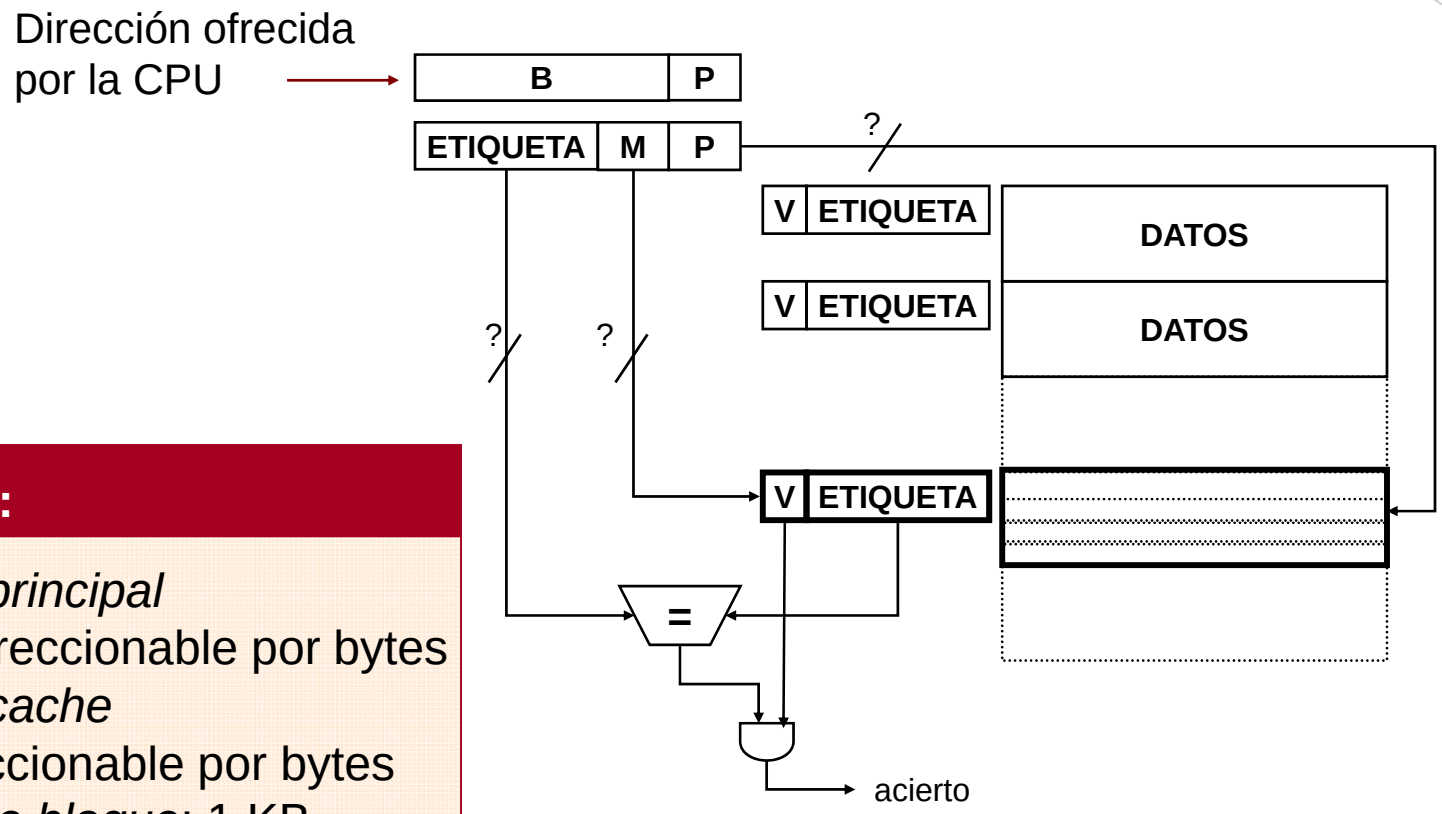
- Cada bloque de la cache puede contener diferentes bloques de memoria
  - ¿Cómo saber si el dato de la cache es el dato buscado?
- Solución:
  - El directorio almacena para cada bloque cache una etiqueta con los n-k-m bits que completan la dirección del bloque almacenado

Emplazamiento directo



- El acceso al bloque cache y al directorio es directo.
- Para conocer si un bloque de mem. ppal está cargado en MC, basta comparar las etiquetas

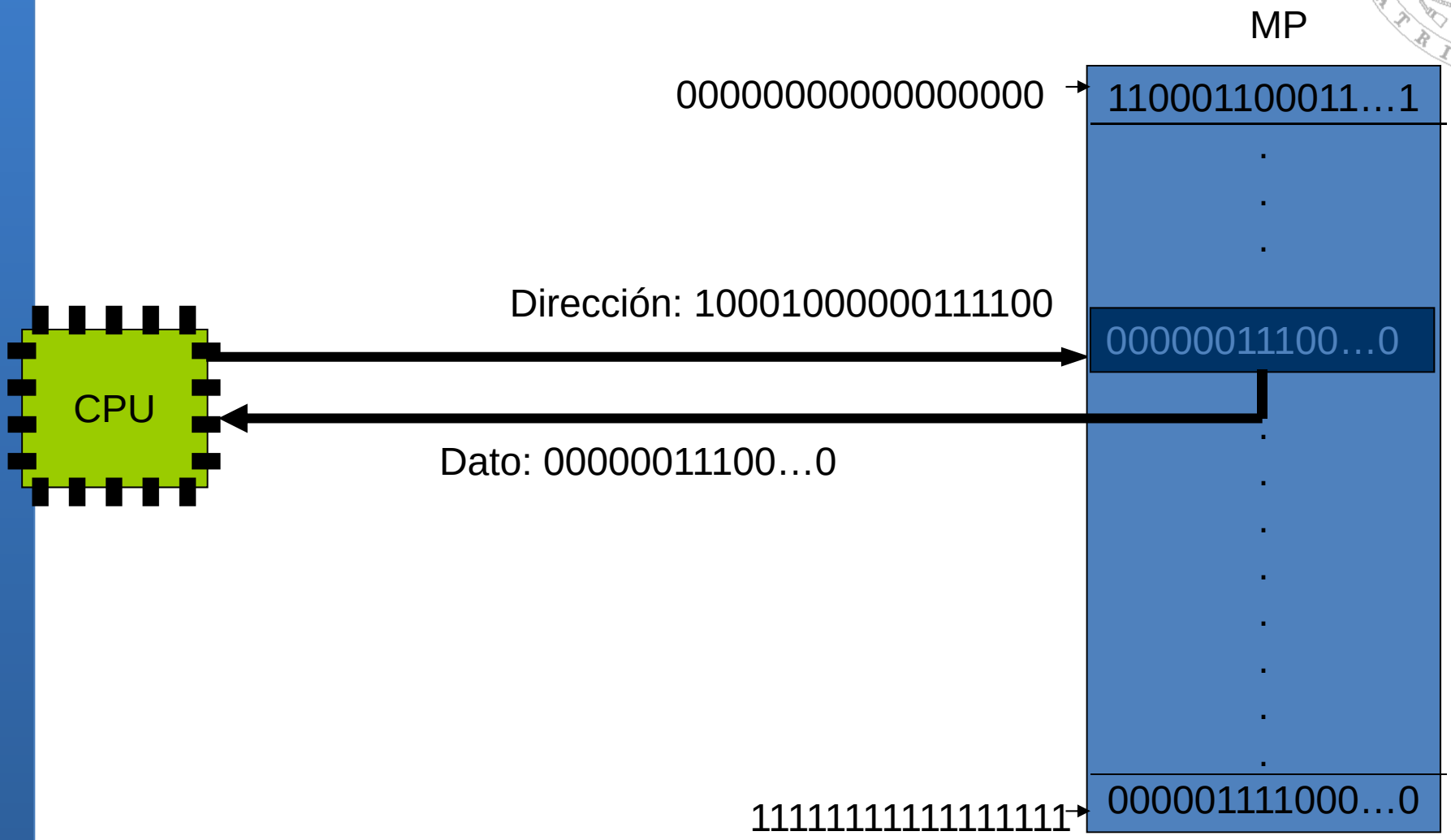
Emplazamiento directo



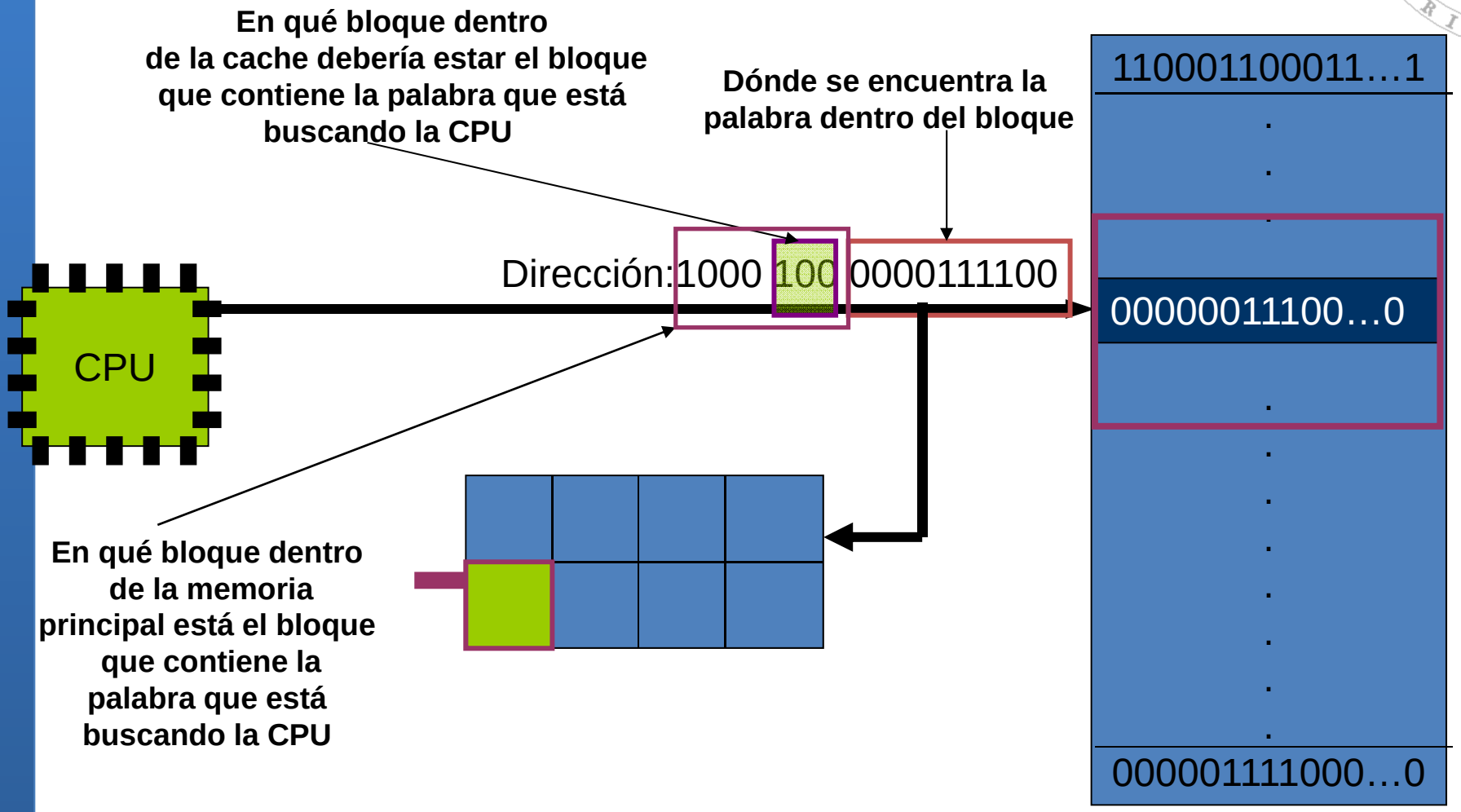
Ejemplo:

Memoria principal
128 KB direccionable por bytes
Memoria cache
8 KB direccionable por bytes
Tamaño de bloque: 1 KB
¿Número de bloques en MP?
¿Número de bloques en MC?

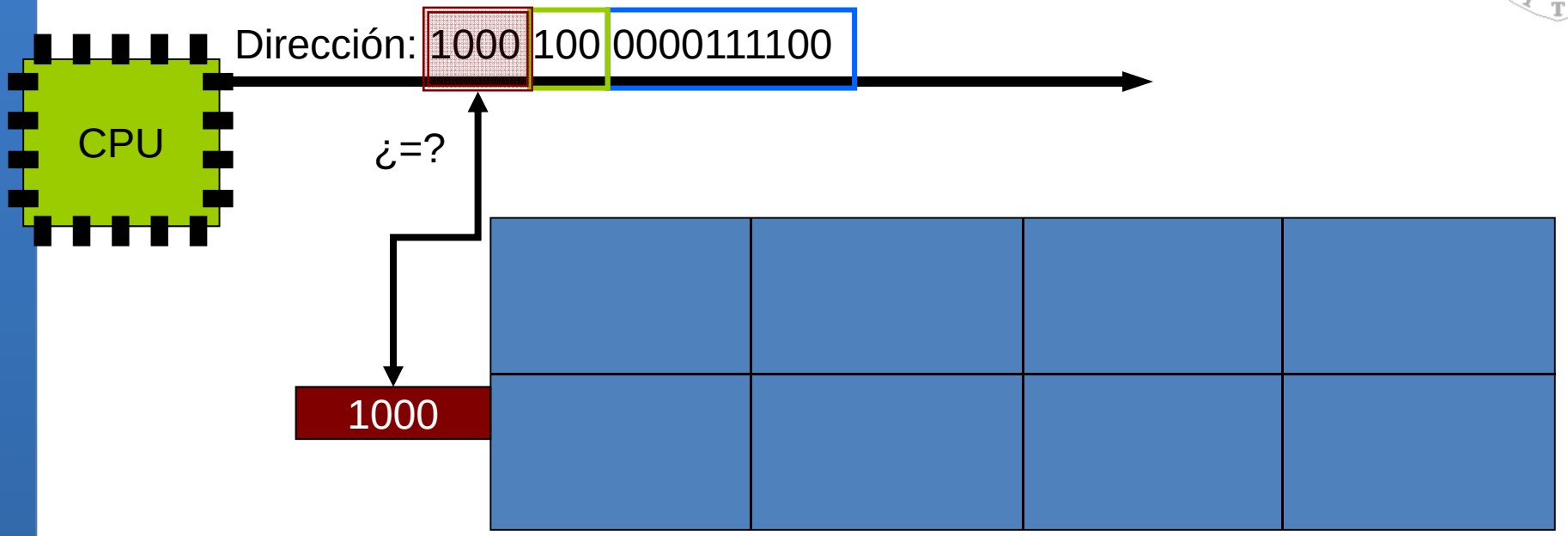
Emplazamiento directo



Emplazamiento directo

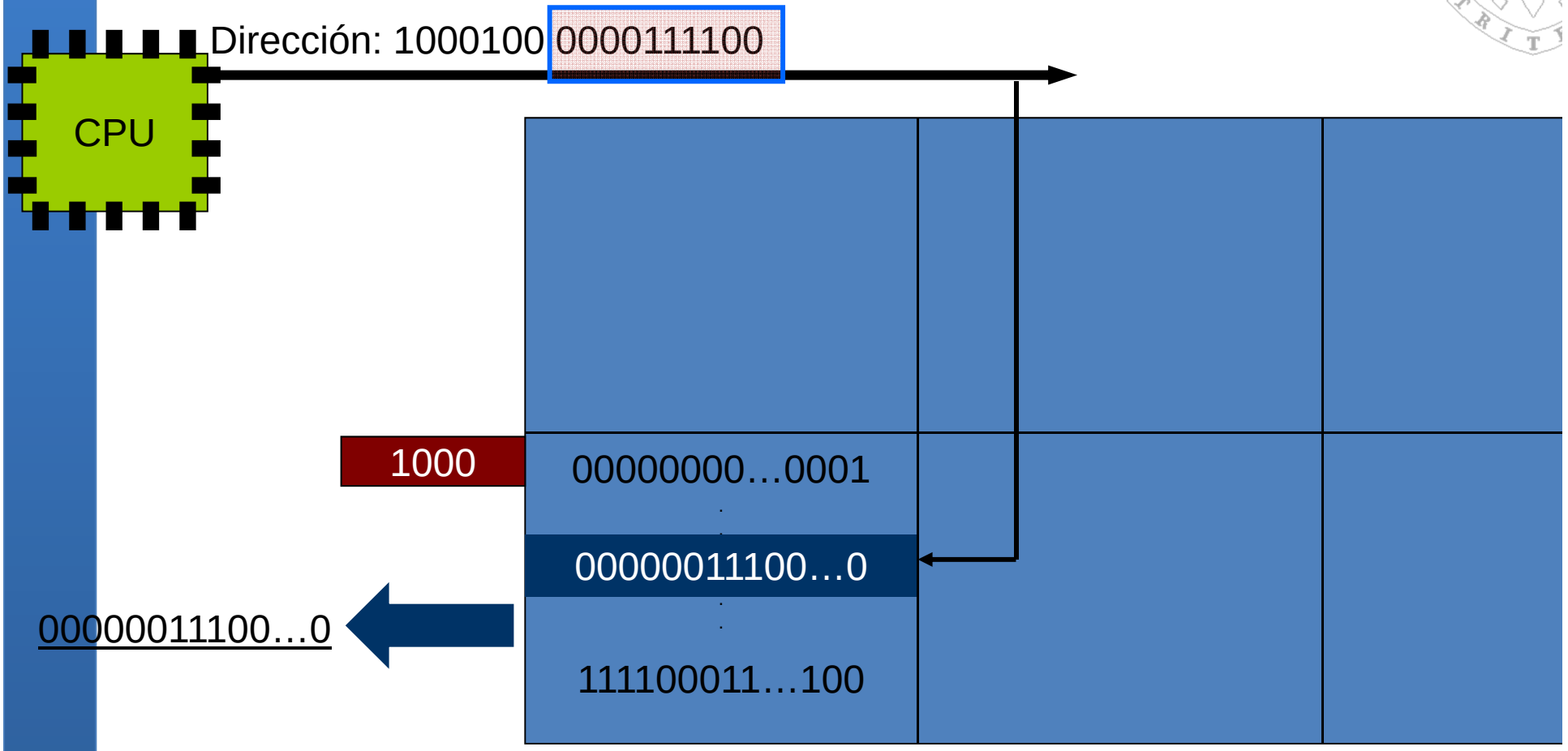


Emplazamiento directo



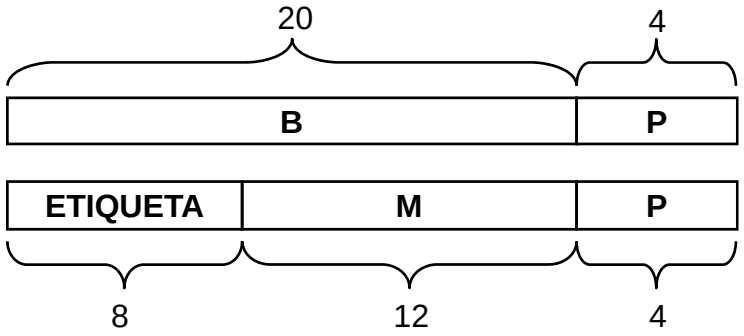
Si las etiquetas son iguales, entonces el bloque guardado en el bloque de cache es el bloque que estamos buscando

Emplazamiento directo



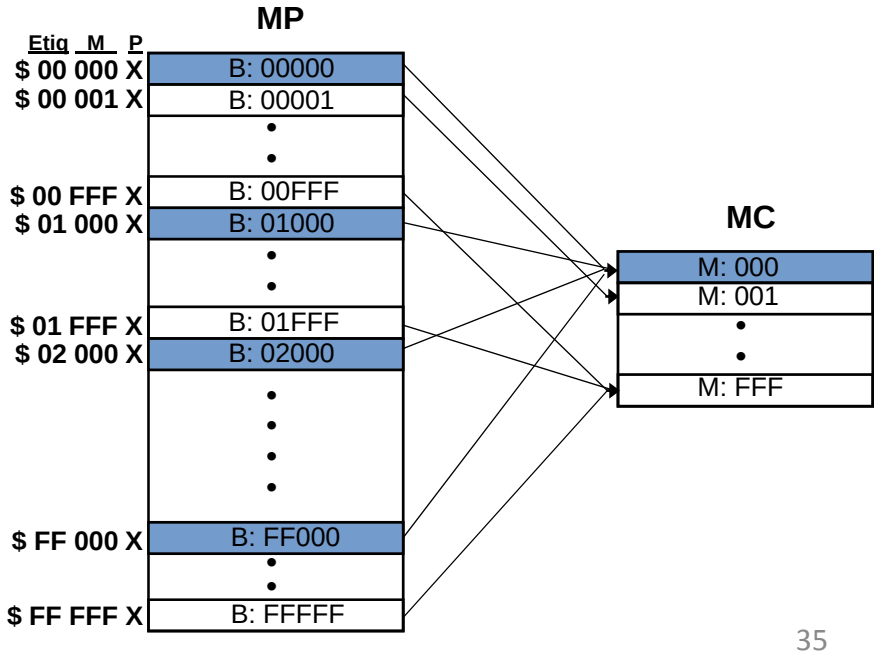
Emplazamiento directo

- *Memoria principal*
  - 16 Mb direccionable por bytes (dirección de 24 bits)
- *Memoria cache*
  - 64 Kb
- *Tamaño de bloque*: 16 bytes
  - Número de bloques en MP
    - nB = 1M (dirección de 20 bits)
  - Número de bloques en MC
    - nM = 4K (dirección de 12 bits)



Direcciones de CPU			***Bloque cache***
XXXXXXXX	000000000000	XXXX	000h
XXXXXXXX	000000000001	XXXX	001h
XXXXXXXX	000000000010	XXXX	002h
.....			.....
XXXXXXXX	111111111110	XXXX	FFEh
XXXXXXXX	111111111111	XXXX	FFFh

Información que se almacena en la etiqueta del bloque cache



Emplazamiento directo

Ejemplo:

- Se tiene una memoria con las siguientes características
  - *Memoria principal*
  - 1 MB direccionable por bytes
  - *Memoria cache*
  - 4 KB direccionable por bytes
  - *Tamaño de bloque*: 1 KB
- Se producen los siguientes accesos
  - 00005
  - 00006
  - 14605
  - 03805
  - 10005
  - 000F5

Emplazamiento directo

Ejemplo:

- 00005
  [0000 0000] [00] 00 0000 0101
  etiqueta m — FALLO!
- 00006
  [0000 0000] [00] 00 0000 0110
  etiqueta m — ACIERTO!
- 14605
  [0001 0100] [01] 10 0000 0101
  etiqueta m — FALLO!
- 03805
  [0000 0011] [10] 00 0000 0101
  etiqueta m — FALLO!
- 10005
  [0001 0000] [00] 00 0000 0101
  etiqueta m — FALLO!
- 000F5
  [0000 0000] [00] 00 1111 0101
  etiqueta m — FALLO!

Emplazamiento directo

Ventajas:

- baja complejidad hardware
- rapidez en la identificación
- directorio pequeño

Principal problema:

Alta tasa de fallos cuando varios bloques compiten por el mismo bloque de MC

Políticas de emplazamiento

Emplazamiento asociativo:

Cada inquilino puede alojarse en **cualquier** piso

Cada bloque de memoria principal puede ubicarse en cualquier bloque cache

Políticas de emplazamiento

Emplazamiento asociativo por conjuntos:

Cada inquilino tiene asignada una **planta** puede ubicarse en **cualquiera piso** de dicha planta

La MC está dividida en nC conjuntos de nM/nC bloques cache cada uno

Cada bloque B tiene asignado un **conjunto fijo** C y puede ubicarse en **cualquiera de los bloques cache** que componen dicho conjunto

fc²

Políticas de actualización

¿Qué sucede cuando la CPU escribe una palabra?

Si se escribe sobre uno de los bloques cargados en la cache, el contenido de la MC y de la MP no coinciden $\Rightarrow$ es necesario actualizar el bloque correspondiente de MP

Política de actualización

- **Escritura inmediata (*write-through*):** cada vez que se hace una escritura en la MC se actualiza inmediatamente la MP.
  - *Ventajas:* bajo coste hardware y consistencia en todo momento
  - *Desventajas:* aumenta el tráfico entre MC-MP

Políticas de actualización

- **Post-escritura (*copy-back*):** la MP se actualiza sólo cuando se reemplaza el bloque
  - Se utiliza un **bit de actualización** por bloque que indica si debe actualizarse en MP.
  - *Ventajas:* disminuye el tráfico entre MC y MP y disminuye el tiempo de acceso para escritura
  - *Desventajas:* inconsistencia
  - Para evitar retrasos en los reemplazamientos se utiliza un buffer intermedio (1 bloque)