2.- Unidad de memoria

1.- Conceptos básicos
2.- Jerarquía de memorias
3.- Memorias a semiconductor
4.- Memorias caché
5.- " asociativas
8.- Discos magnéticos

1.- Conceptos básicos

Aspectos de las memorias
- Localización
- Capacidad
- Unidad de transferencia
- Método de acceso
- Tipos físicos
- Características físicas
- Velocidad
- Organización

Localización
- Interna → Regist. internos → { - Pequeña, - Rápida }
- Principal → { central o primaria } → Relativamente { - Grande, - Rápida }
- Secundaria-externa → Acceso mediante E/S ⇒ { - Controlador + - Soporte }
  - Más grande y lenta que la principal

Capacidad ⇒ Nº bits - bytes - palabras

Capacidad = Nº palabras × Nº bit por palabra

Unidad de transferencia = Nº bits que se transfieren en cada bloque de transferencia

En la mem. principal la unidad de transferencia es la anchura del bus de datos.

Conceptos
- Palabra = Unidad natural de organización de mem.
- Unidades direccionales = Nº bit / dirección
  Normalmente : Palabra = Unid. direccio
- Unidad de transferencia = Nº bits leidos/escritos simultáneamente en memoria
  En mem. externas → bloques
  En discos → sectores / clusters

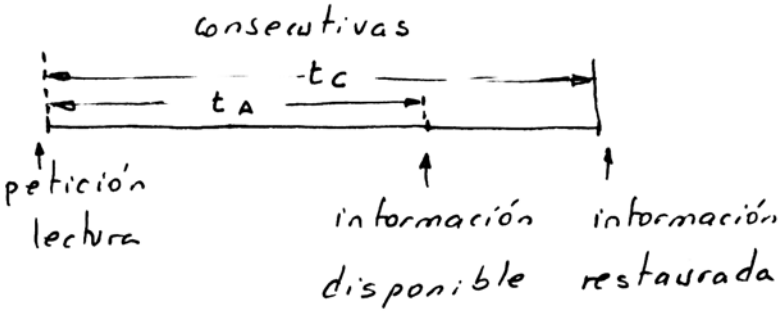
Acceso
- Aleatorio : acceso en cualquier orden.
  Cada dato asociado a una dirección
  t. acceso = cte e independiente a la secuencia acceso
- Secuencial : acceso mediante secuencia lineal.
  Unidad de datos elemental = registro
- Directo : combinación aleatorio + secuencial
  Discos ⇒ aproximación + búsqueda final
- Asociativo : Son aleatorias ; pero el parámetro de búsqueda NO es una dirección si no un campo específico del contenido.

Tipos físicos
- Mem. a semiconductor
- Magnéticas
- Ópticas
- Magneto-ópticas

Características físicas
- Alterabilidad = Modificación contenido
  - RAM
  - ROM
  - PROM
  - EPROM
    - EEPROM
    - UVEPROM
- Permanencia
  - Lectura destructiva
  - Volatilidad
  - Almacenamiento
    - Estático
    - Dinámico

Velocidad: En función del tiempo en acceder a la información

- tiempo acceso: tiempo en leer/escribir una palabra (t_A)
- tiempo de ciclo de memoria: tiempo entre dos lecturas consecutivas (t_C)



- velocidad de transferencia = Nº palabras/segundo (f_A)

Acceso aleatoria ⇒ $f_A = \frac{1}{t_C}$

Acceso no aleatoria ⇒ $t_n = t_A + \frac{n}{p}$

- t_n = t. R/W n bits
- t_A = t. acces. medio
- n = nº bits trans
- p = veloc. transfer.

UM. 3

Organización: la disposición física de los bits para formar palabras

Tipos { 2 D
2½ D

2.- Jerarquía de memorias

Parámetros en la jerarquía de memorias { - Capacidad, - Velocidad, - Coste } Compromiso entre ellas

↓ por unidad de almacenamiento

Jerarquía

Capac. ↓ Veloc. ↑ Coste ↑

Nivel 0 - Registros CPU

" 1 - Mem. caché

" 2 - Mem. principal

" 3 - Discos magnéticos

- cintas magnéticas

Mem. interna

Mem. externa

Frecuencia acceso ↑

Funcionamiento: información pasa de nivel 2 a nivel 1 antes de ser usada por la CPU

$$t_A = t_{A1} + t_{A2} - \frac{T \, t_{A2}}{100}$$

t_{A1}: t. acceso nivel 1

t_{A2}: " " " 2

T = % del tiempo total en el que la palabra está en el nivel 1

- $T=0 \Rightarrow t_A = t_{A1} + t_{A2} \Rightarrow$ todas palabras en niv. 2
- $T=100\% \Rightarrow t_A = t_{A1} \Rightarrow$ " " " " 1

Principio de localidad: se entiende como el índice de probabilidad de uso de la información que está en memoria.

- Localidad temporal: Tendencia a reutilizar los datos e instrucciones utilizadas recientemente.
  - BUCLES
- Localidad espacial: Tendencia a referenciar las instrucciones y datos próximos a los que se están utilizando.

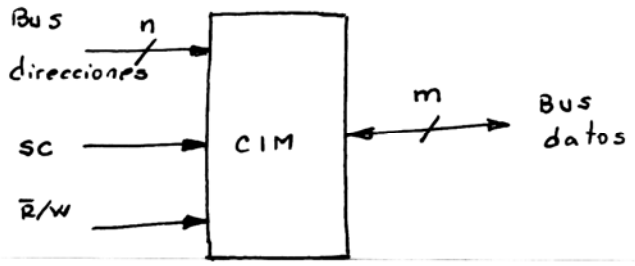
3.- Memorias a semiconductor

CIM = Circuitos Integrados de Memoria

Organización interna → matriz de "N x m" celdas elementales

N = nº palabras

m = bits en cada palabra



n = bus direcciones

m = " datos

R/W = lectura / escritura

SC = chip Select

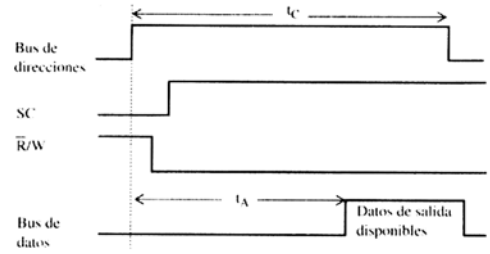


Figura 2.10: Ciclo de lectura de un CIM

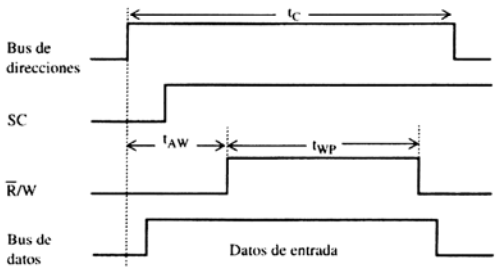


Figura 2.11: Ciclo de escritura de un CIM

t_C = tiempo de ciclo

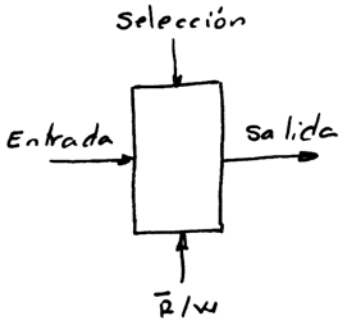
t_A = tiempo acceso

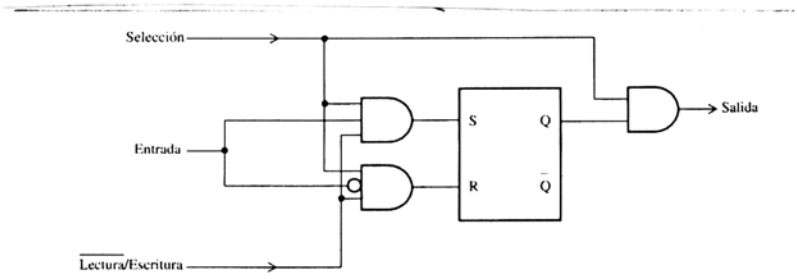
t_{AW} = tiempo de fijación de dirección

t_{WP} = anchura de pulso de escritura

$t_W = t_{AW} + t_{WP}$

Estructura de la celda básica de memoria (RAM)





Organización interna → Tipos { 2D, 2½D }

2D → {
- Realización física → coste elevado
- organización por palabras o lineal
- Rápida
- Matrices excesivamente largas y estrechas
}

Limitaciones {
- Complejidad del decodifica. direcciones
- Nº de líneas de dirección y datos
}

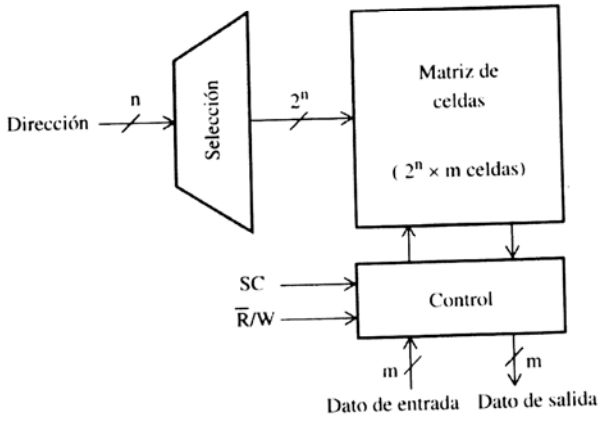


Figura 2.13: Organización 2D de una memoria RAM

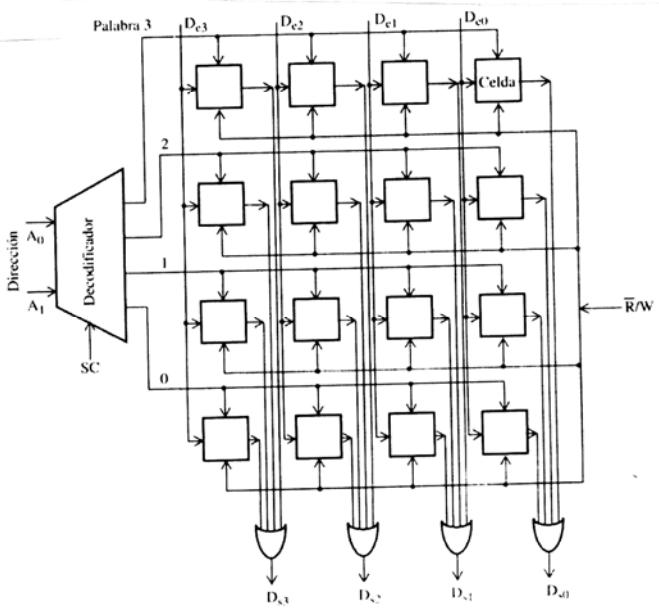
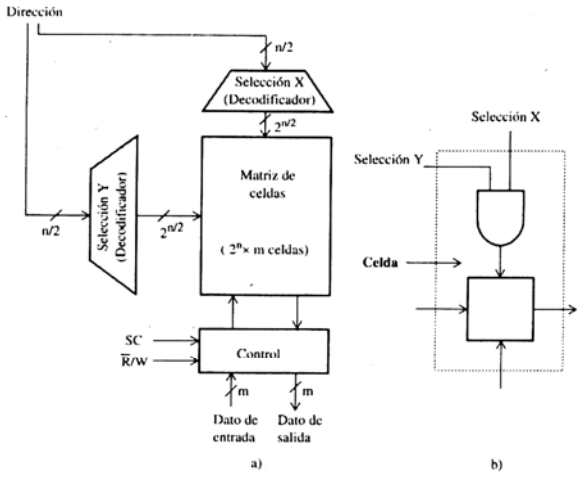


Figura 2.14: Memoria RAM de 4 palabras con 4 bits por palabra

Organización 2½ D

Formada por celdas de memoria de forma que con parte del bus de direcciones seleccionamos la celda y con la otra parte la linea de dicha celda.



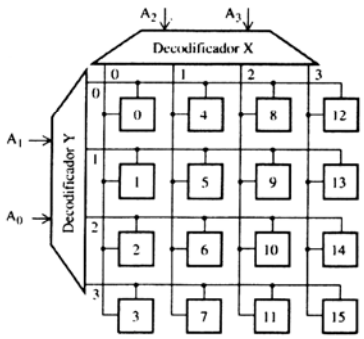


Figura 2.15: a) Memoria RAM con decodificación por coincidencia b) Celda básica de memoria modificada

Figura 2.16: Memoria RAM de 16 × 1 con selección por coincidencia

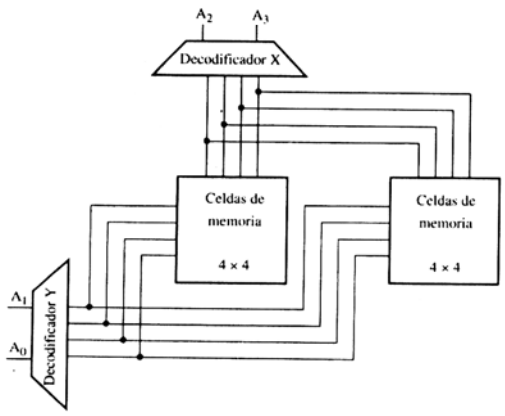
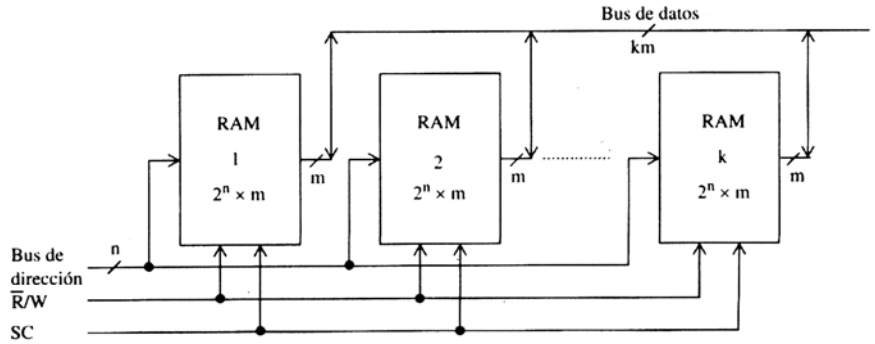


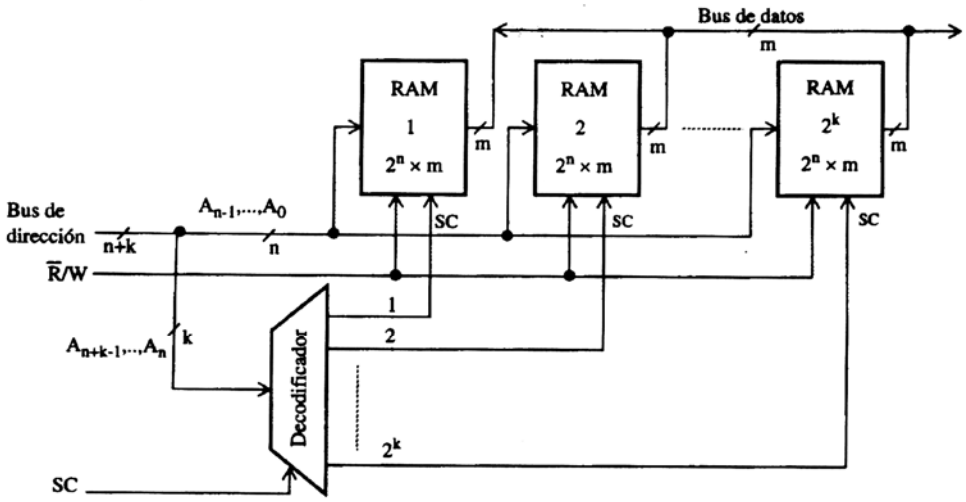
Figura 2.19: Memoria RAM de 16 × 2 con selección por coincidencia

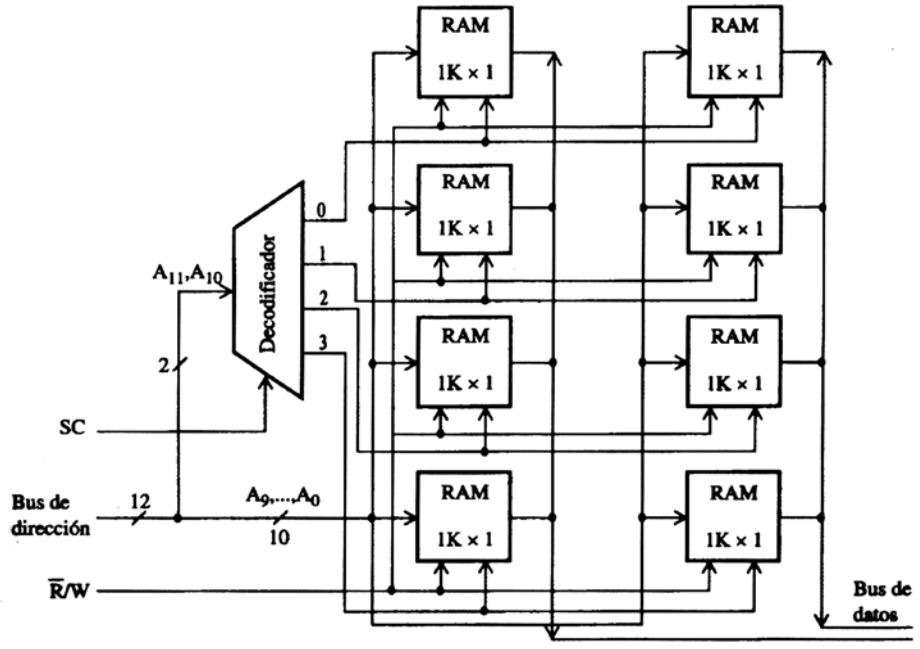
UM. 8

- Diseño de bloques de memoria

Objetivos { - Incrementar tamaño de las palabras
- " nº de palabras







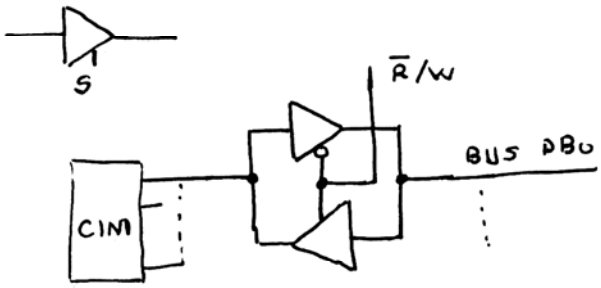
UM.9

- Conexión de la unidad de memoria al bus del sistema

Tipos buses
- Unidireccionales → Direcciones y Control
- Bidireccionales → Datos

Unidireccionales ⇒

Bidireccionales ⇒



- Estructura y direccionamiento de la unidad de memoria

Memoria principal ⇒ Conjunto de bloques de memoria iguales y apilables

Ventajas
- Abaratamiento de costes
- Modularidad y posibilidad de ampliación
- Fácil mantenimiento y reparación

Asignación de espacio mediante personalización

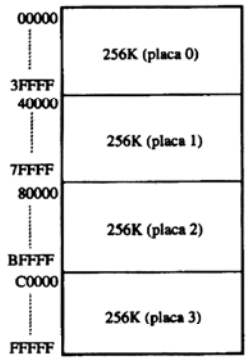
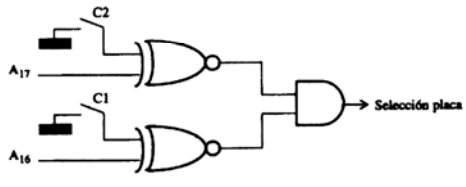


Figura 2.29: Mapa de memoria



4. Memorias caché

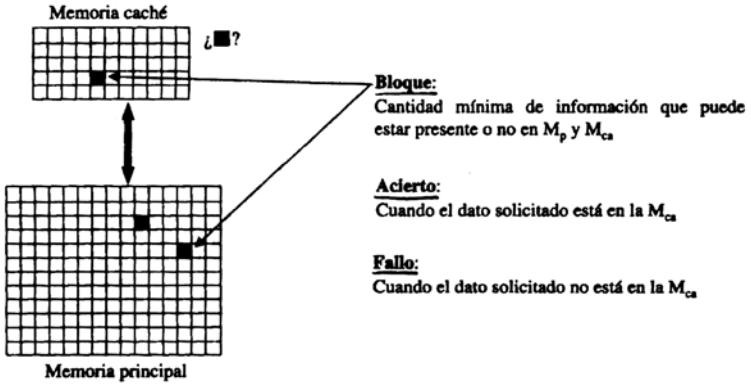
Mem. caché = mem. pequeña y rápida colocada entre la mem. principal y la CPU

t. ciclo mem. > t. ciclo del procesador ⇒ CPU tiene que esperar a la mem. en los accesos a ésta

⇓

Mem. caché acela el proceso

- Conceptos



- Organización

Mem. ppal (Mp) ⇒ n bits bus dir ⇒ 2^n palabras

⇘ M bloques de k palabras por bloque

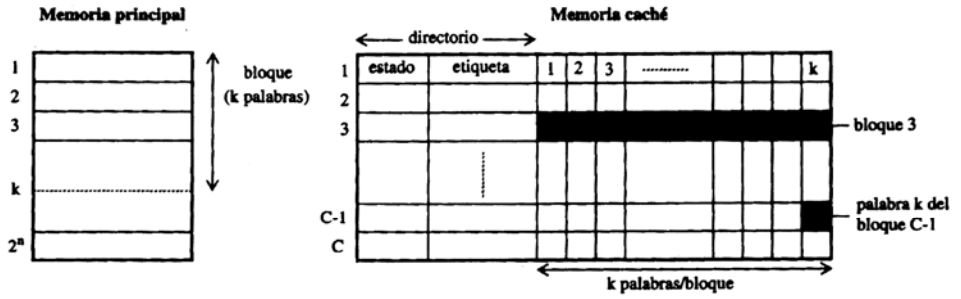
⇓

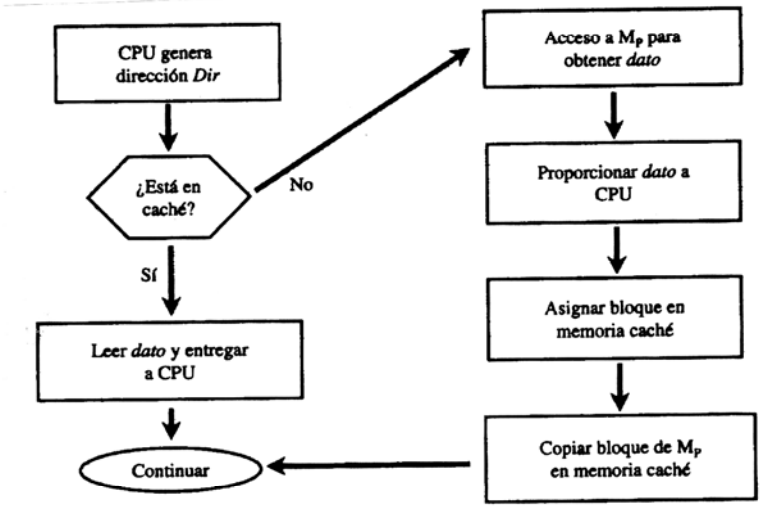
$$M = \frac{2^n}{k}$$

Caché (Mca) ⇒ C particiones de k palabras cada una

En cada dirección una etiqueta indicativa de la dirección

$$\boxed{M >> C}$$





Criterios de diseño

- Capacidad : 1K, 4K, 16K, 32K, ...
- Organización
  - Directa
  - Totalmente asociativa
  - Asociativa por conjuntos
- Mecanismo búsqueda
  - Por demanda
  - Con anticipación
  - Selectiva
- Algorit. reemplazamiento
  - Utiliz. menos reciente (LRU)
  - Más antigua (FIFO)
  - Utiliz. menos frecuente (LFU)
  - Aleatorio
- Estrategia escritura
  - Escritura inmediata
  - Post-escritura
  - Escritura única
- Tamaño bloque : 4, 8, 16, 32.. palabras
- Numero cachés : Nº niveles : 1, 2, ...

- Rendimiento de una memoria caché

tasa de acierto (h) = $\frac{\text{Nº veces que solicitud en Mca}}{\text{Nº total solicitudes}}$ ← aciertos

$h = \frac{\text{Nº aciertos}}{\text{Nº aciertos + fallos}}$

PRINCIPIO DE LOCALIDAD

diseño efectivo ⇒ $h > 0{,}9$

tasa de fallos = $1 - h$

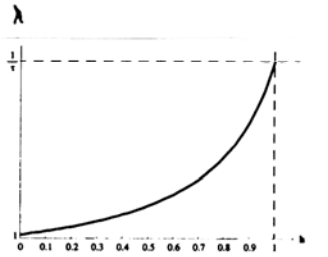
t. acceso medio $ta = h \cdot tca + (1-h)\, tp$

tca = t. acc. med. Mca

tp = t. acc. med. MP

T = razón entre tca y tp ⇒ $T = \frac{tca}{tp}$ $\quad 0{,}1 < T < 0{,}5$

índice de mejora $\lambda = \frac{tp}{ta} = \frac{1}{1 - h(1-T)}$



...ndice de mejora λ de la M_{ca} en función de la tasa de acierto h

- Capacidad de la mem. caché

Grande ⇒
- Lógica más compleja
- Más lentas
- Mayor espacio físico

⇓

Equilibrio ⇒ 1K ÷ 512K

ORGANIZACIÓN DE LA MEMORIA CACHÉ

Establecer la función de correspondencia que asigna a los bloques de la memoria principal en las posiciones definidas en la memoria caché

Técnicas:	Directa Totalmente asociativa Asociativa por conjuntos

Parámetros del ejemplo a utilizar en las descripciones:

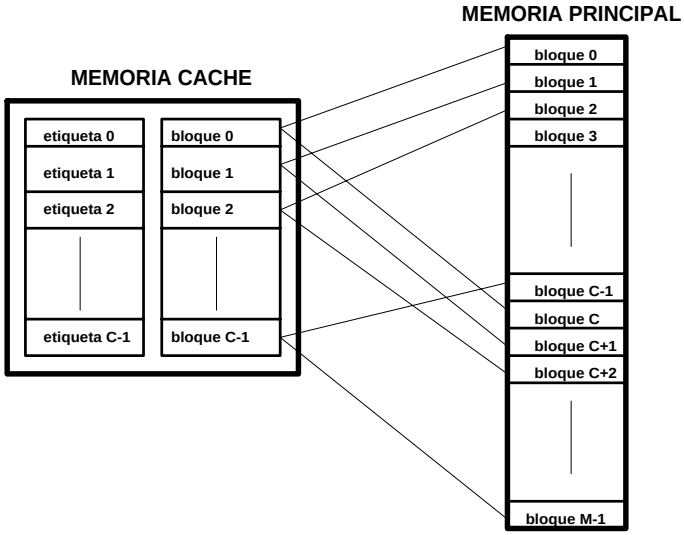
a) Ancho de palabra de datos 16 bits
b) Memoria caché 512 B = 2^9 Bytes
c) Tamaño de bloque k = 8
d) Memoria principal = 32 KB

Consecuencias:

- 32 KB = 2^{15} $\Rightarrow$ Bus de direcciones = 15 bits $\Rightarrow$ A0 a A14
- 512B = 2^9 $\Rightarrow$ Bus direcciones de la caché $\Rightarrow$ 9 bits
- 512B y k = 8 $\Rightarrow$ Nº bloques = 512/8 = 64 bloques
- k = 8 $\Rightarrow$ 2^3 $\Rightarrow$ 3 bits
- 64 bloques $\Rightarrow$ 2^6 $\Rightarrow$ 6 bits

CORRESPONDENCIA DIRECTA

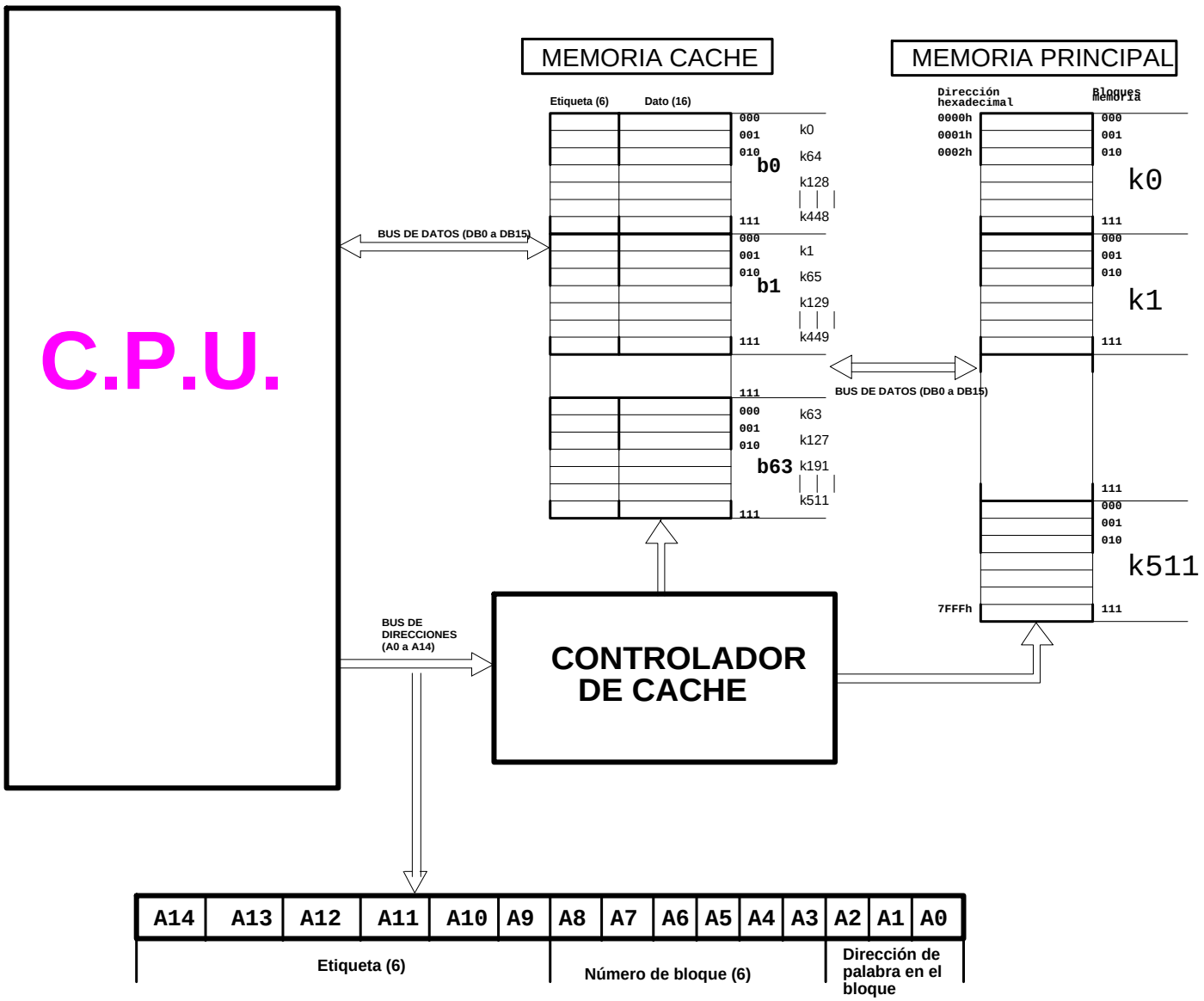
Cada bloque de la memoria principal tiene su posición en la caché y ***SIEMPRE*** en el mismo sitio



Ventajas	Simple Económica

Inconveniente: Cada bloque tiene asignada una posición fija en la memoria caché $\Rightarrow$ ante continuas referencias a palabras de dos bloques con la misma localización en caché, continuos fallos habiendo sitio libre en la caché

Esquema de la caché



Funcionamiento:

Principios:

- Palabras por bloque 8 ⇒ 3 bits (b0 a b2)
- Nº bloques 64 ⇒ 6 bits
- Etiqueta = (bus direcc) – (bits de bloques) -- (bits palab/bloque) ⇒ 15 – 6 – 3 = 6
- Ancho de la mem. caché = Ancho palabra + ancho etiqueta ⇒ 16 + 6 = 22 bits

1. La CPU entrega la dirección de mem. al controlador de caché.
2. El ctrl. de caché aísla de los bits b3 a b8 y con ellos apunta al bloque correspondiente.
3. Comprueba si la etiqueta en caché = bits correspondientes a etiqueta en bus de direcciones (b9 a b14)
   - 3.a Si iguales ⇒ acierto, con lo que coge de los bits b15 a b0 (dato) de la caché y los saca al bus de datos de la CPU.
   - 4.a Fin de acceso.

   - 4.b Si distintos ⇒ Fallo
   - 5.b El ctrl. de caché transmite bloque completo desde mem. ppal. a mem. caché
     - 5.b.1 El ctrl. Caché pone a 000 los bits de dir. de palabra (b2 a b0).
     - 5.b.2 El ctrl. Caché apunta al bloque indicado por bus dir. (b3 a b8).
     - 5.b.3 El ctrl. Caché lleva de la dir. b14 b13 b12 .....b3 0 0 0 de mem. ppal. a la caché al bloque formado por los bits b8 a b3 en la dir. de palabra del bloque 0 0 0 (palabra 000 del bloque). Incrementa un contador y lleva de la dir. b14 b13 b12 .....b3 0 0 1 de mem. ppal. a la caché al bloque formado por los bits b8 a b3 en la dir. de palabra del bloque 0 0 1 (palabra 001 del bloque). Así sucesivamente hasta la palabra b14 b13 b12 .....b3 1 1 1 (palabra 7 del bloque) . Escribiendo en la zona de la etiqueta el valor de los bits b14 a b9.
     - 5.b.4 Una vez ha cargado el bloque completo (8 palabras) ⇒ Hay acierto y se salta al paso 4-a.

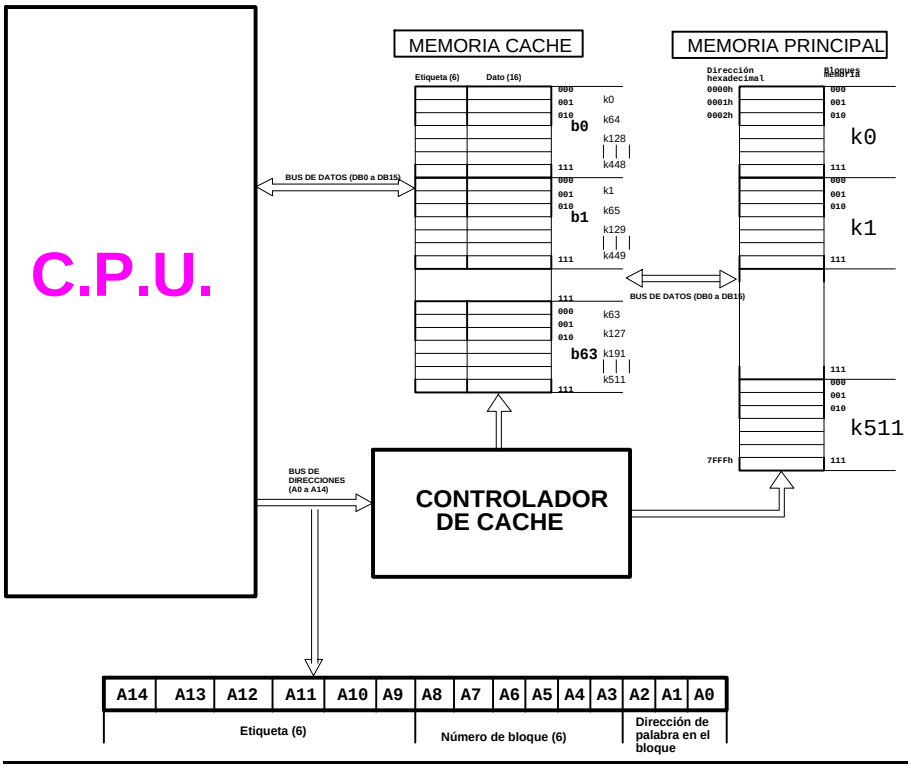
Ejemplo con los parámetros indicados

- Se supone que todavía no ha habido ningún acceso ⇒ Mem. caché vacía ⇒ todo con 0
- Se quiere acceder al contenido de la posición mem. $025F_H$

$025F_H$ =

b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	1	0	0	1	0	1	1	1	1	1
etiqueta						**Nº bloque**						**Dir. palabra**		

Bits dir. palabra = 3	Bit bloque = 6	Bits etiqueta = 6	Ancho palabra = 16



- El ctrl. de la caché mira si la etiqueta del bloque:

0	0	1	0	1	1

Es:

0	0	0	0	0	1

Como es el primer acceso la mem. caché contiene todo 0 ⇒ hay primer fallo.

- El ctrl. de caché va a direccionar a la mem. ppal. a la dirección

0	0	0	0	0	1	0	0	1	0	1	1	0	0	0

y llevará el contenido al bloque

0	0	1	0	1	1

dirección

0	0	0

escribiendo la etiqueta

0	0	0	0	0	1

Repitiendo el proceso hasta:

- El ctrl. de caché va a direccionar a la mem. ppal. a la dirección

0	0	0	0	0	1	0	0	1	0	1	1	1	1	1

y llevará el contenido al bloque

0	0	1	0	1	1

dirección

1	1	1

escribiendo la etiqueta

0	0	0	0	0	1

- Con lo que el bloque completo estará escrito en la mem. caché , existiendo ahora acierto y presentando en el bus de datos el contenido de la mem. caché correspondiente al bloque

0	0	1	0	1	1

Y la dirección de palabra

1	1	1

- Si posteriormente se quiere acceder al contenido de la dirección de mem. $085B_H$

$085B_H$

b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	1	0	0	0	0	1	0	1	1	0	1	1
etiqueta						**Nº bloque**						**Dir. palabra**		

Irá al bloque

0	0	1	0	1	1

Que estás ocupado con el dato anterior de la dirección $025F_H$

Por lo tanto la comparación de la etiqueta de la caché con la parte correspondiente a la etiqueta del bus de direcciones presentará desigualdad $\Rightarrow$ Fallo

Etiqueta de la caché

0	0	0	0	0	1

bits correspondientes a etiqueta del bus de direcciones

0	0	0	1	0	0

Por lo que llevará el bloque desde la dirección de mem. ppal.

0	0	0	1	0	0	0	0	1	0	1	1	0	0	0

a la dirección

0	0	0	1	0	0	0	0	1	0	1	1	1	1	1

al bloque

0	0	1	0	1	1

de la caché, existiendo actualmente un acierto y presentando en el bus de datos el contenido de la mem. caché correspondiente al bloque

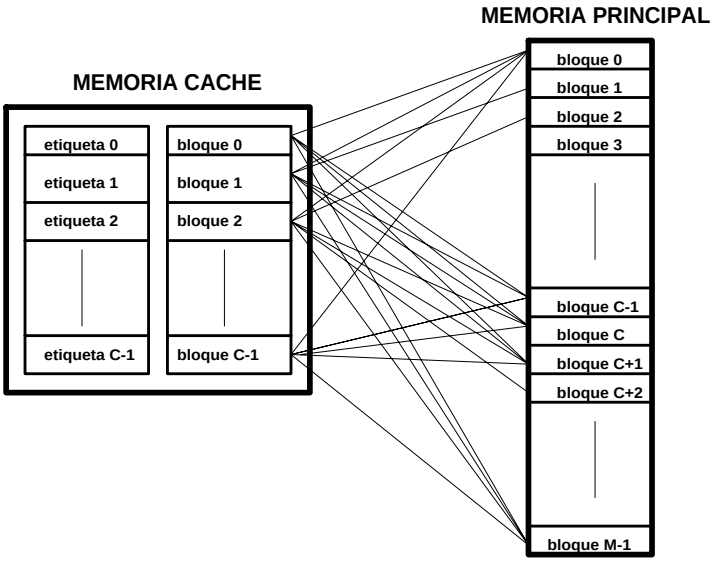
0	0	1	0	1	1

Y la dirección de palabra

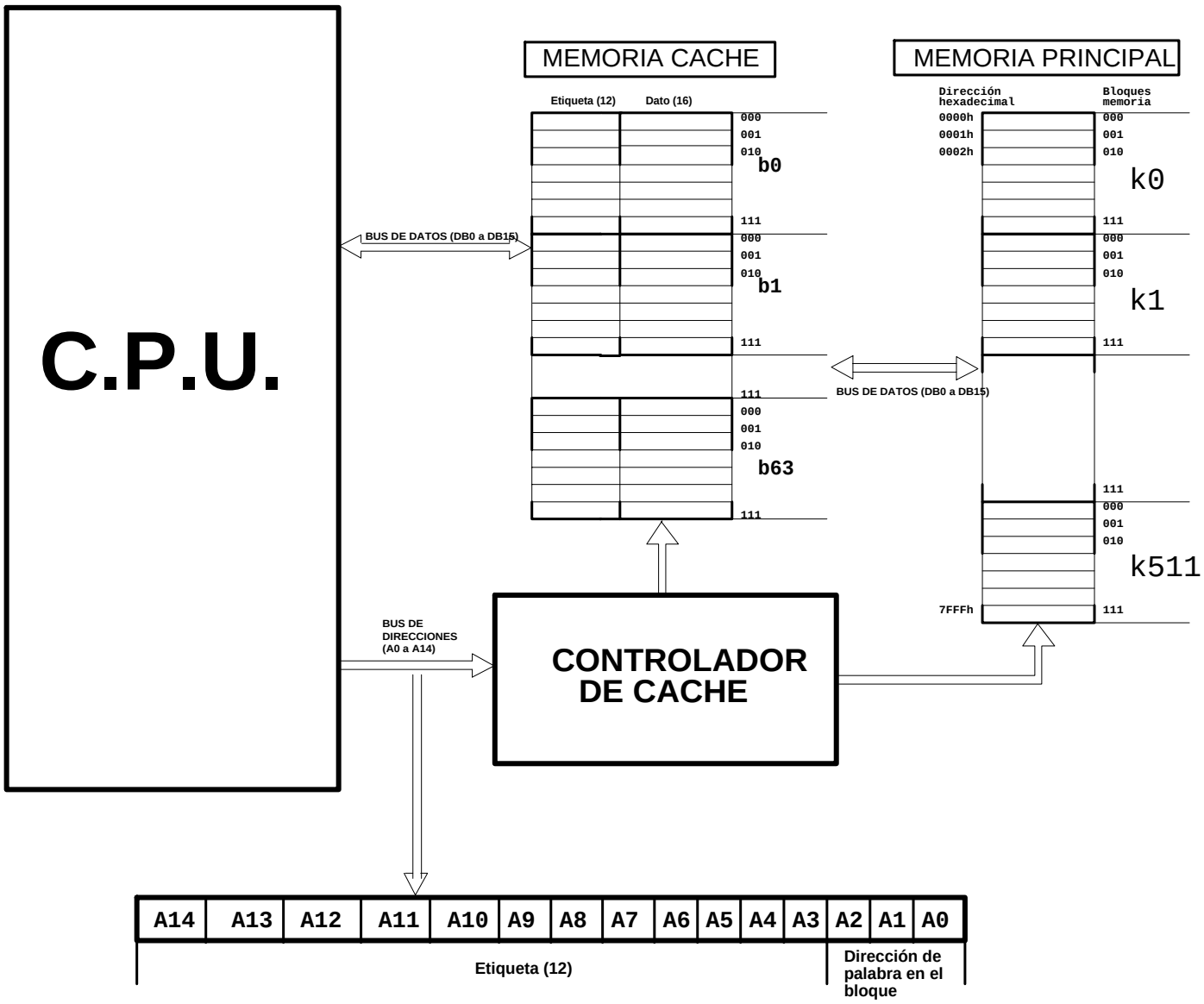
0	1	1

CORRESPONDENCIA TOTALMENTE ASOCIATIVA

Los bloques de la mem principal se alojan en cualquier bloque de la mem caché, comprobando solamente la etiqueta de todos y cada uno de los bloques para verificar acierto.
El principal inconveniente es que precisa una circuitería compleja para hacer la comparación paralelo de todos los campos de etiqueta.



Esquema de la caché



Funcionamiento:

Principios:

- Palabras por bloque 8 ⇒ 3 bits (b0 a b2)
- Nº bloques 64
- Etiqueta = (bus direcc) – (bits palab/bloque) ⇒ 15 –3 = 12
- Ancho de la mem. caché = Ancho palabra + ancho etiqueta ⇒ 16 + 12 = 28 bits

1. La CPU entrega la dirección de mem. al controlador de caché.
2. El ctrl. de caché busca en todos y cada uno de los bloques coincidencia entre los bits b15 a b3 del bus de direcciones con la etiqueta.

3.a Si encuentra coincidencia ⇒ acierto, con lo que coge de los bits b15 a b0 (dato) del bloque de la caché con coincidencia y los saca al bus de datos de la CPU.

4.a Fin de acceso.

4.b Si no encuentra coincidencia ⇒ Fallo

5.b El ctrl. de caché transmite bloque completo desde mem. ppal. a mem. caché

5.b.1 El ctrl. Caché busca un bloque de caché para librarlo (algoritmo de reemplazamiento).

5.b.2 Una vez liberado el ctrl. Caché pone a 000 los bits de dir. de palabra (b2 a b0).

5.b.2 El ctrl. Caché apunta al bloque liberado.

5.b.3 El ctrl. Caché lleva de la dir. b14 b13 b12 .....b3 0 0 0 de mem. ppal. a la caché al bloque liberado en la dir. de palabra del bloque 0 0 0 (palabra 000 del bloque). Incrementa un contador y lleva de la dir. b14 b13 b12 .....b3 0 0 1 de mem. ppal. a la caché al bloque liberado en la dir. de palabra del bloque 0 0 1 (palabra 001 del bloque). Así sucesivamente hasta la palabra b14 b13 b12 .....b3 1 1 1 (palabra 7 del bloque) . Escribiendo en la zona de la etiqueta el valor de los bits b14 a b3.

5.b.4 Una vez ha cargado el bloque completo (8 palabras) ⇒ Hay acierto y se salta al paso 3.a.

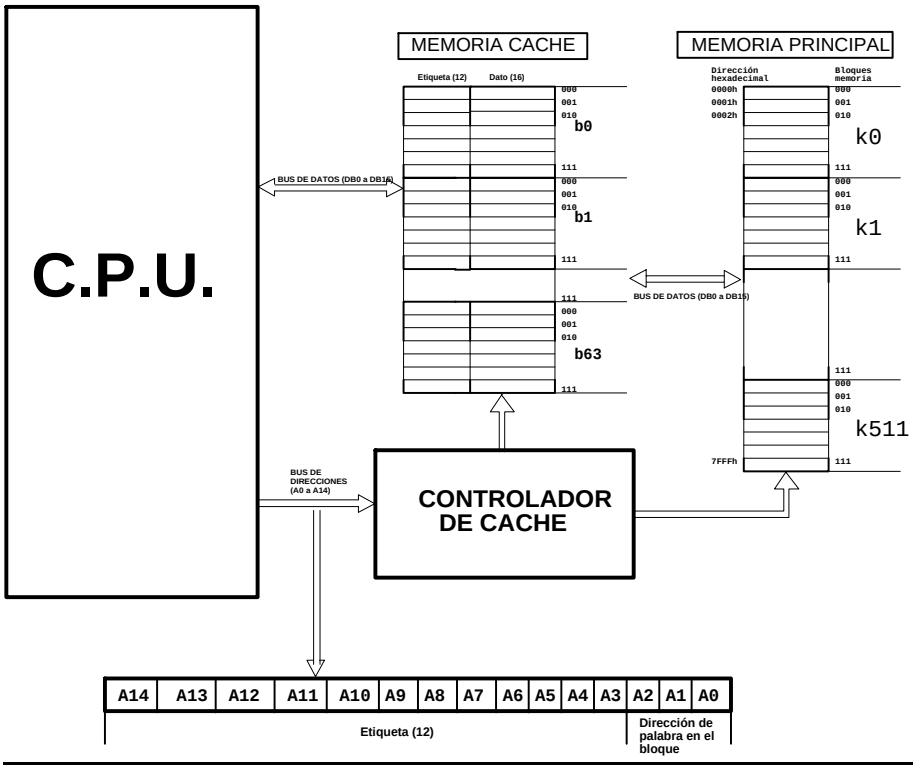
Ejemplo con los parámetros indicados

- Se supone que todavía no ha habido ningún acceso ⇒ Mem. caché vacía ⇒ todo con 0
- Se quiere acceder al contenido de la posición mem. $025F_H$

$025F_H$ =

b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	1	0	0	1	0	1	1	1	1	1
etiqueta												**Dir. palabra**		

Bits dir. palabra = 3	Bits etiqueta = 12	Ancho palabra = 16



- El ctrl. de la caché mira si encuentra la etiqueta en alguno de los bloques:

0	0	0	0	0	1	0	0	1	0	1	1

Como es el primer acceso la mem. caché contiene todo 0 $\Rightarrow$ hay primer fallo.

- El ctrl. de caché va a direccionar a la mem. ppal. a la dirección

0	0	0	0	0	1	0	0	1	0	1	1	0	0	0

y llevará el contenido al bloque primero

0	0	0	0	0	0

dirección

0	0	0

escribiendo la etiqueta

0	0	0	0	0	1	0	0	1	0	1	1

Repitiendo el proceso hasta:

El ctrl. de caché va a direccionar a la mem. ppal. a la dirección

0	0	0	0	0	1	0	0	1	0	1	1	1	1	1

y llevará el contenido al bloque

0	0	0	0	0	0

dirección

1	1	1

escribiendo la etiqueta

0	0	0	0	0	1	0	0	1	0	1	1

- Con lo que el bloque completo estará escrito en la mem. caché , existiendo ahora acierto y presentando en el bus de datos el contenido de la mem. caché correspondiente al bloque

0	0	0	0	0	0

Y la dirección de palabra

1	1	1

- Si posteriormente se quiere acceder al contenido de la dirección de mem. $085B_H$

$085B_H$

b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	1	0	0	0	0	1	0	1	1	0	1	1
etiqueta												**Dir. palabra**		

El ctrl. de la caché mira si encuentra la etiqueta en alguno de los bloques, que no encontrará porque todavía no ha sido cargado $\Rightarrow$ Fallo

- El ctrl. de caché busca un bloque libre

0	0	0	0	0	1

- El ctrl. de caché va a direccionar a la mem. ppal. a la dirección

0	0	0	1	0	0	0	0	1	0	1	1	0	0	0

y llevará el contenido al bloque segundo

0	0	0	0	0	1

dirección

0	0	0

escribiendo la etiqueta

0	0	0	1	0	0	0	0	1	0	1	1

Repitiendo el proceso hasta:

El ctrl. de caché va a direccionar a la mem. ppal. a la dirección

0	0	0	1	0	0	0	0	1	0	1	1	1	1	1

y llevará el contenido al bloque

0	0	0	0	0	1

dirección

1	1	1

escribiendo la etiqueta

0	0	0	1	0	0	0	0	1	0	1	1

- Con lo que el bloque completo estará escrito en la mem. caché , existiendo ahora acierto y presentando en el bus de datos el contenido de la mem. caché correspondiente al bloque

0	0	0	0	0	1

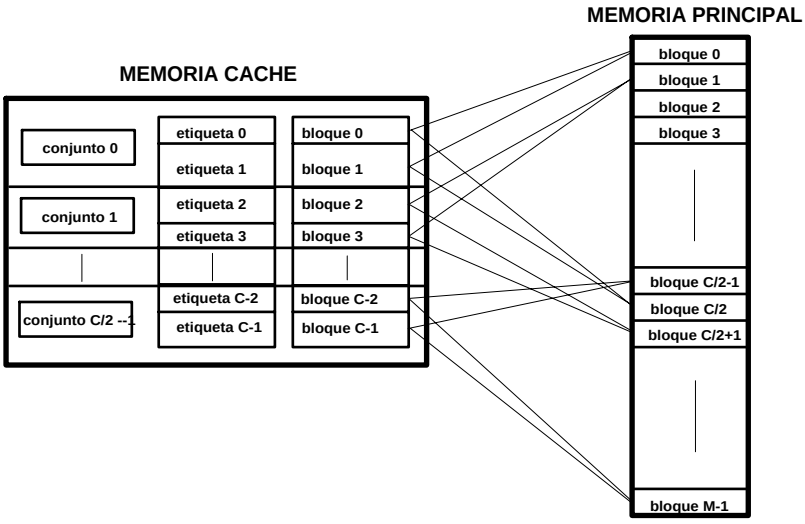
Y la dirección de palabra

0	1	1

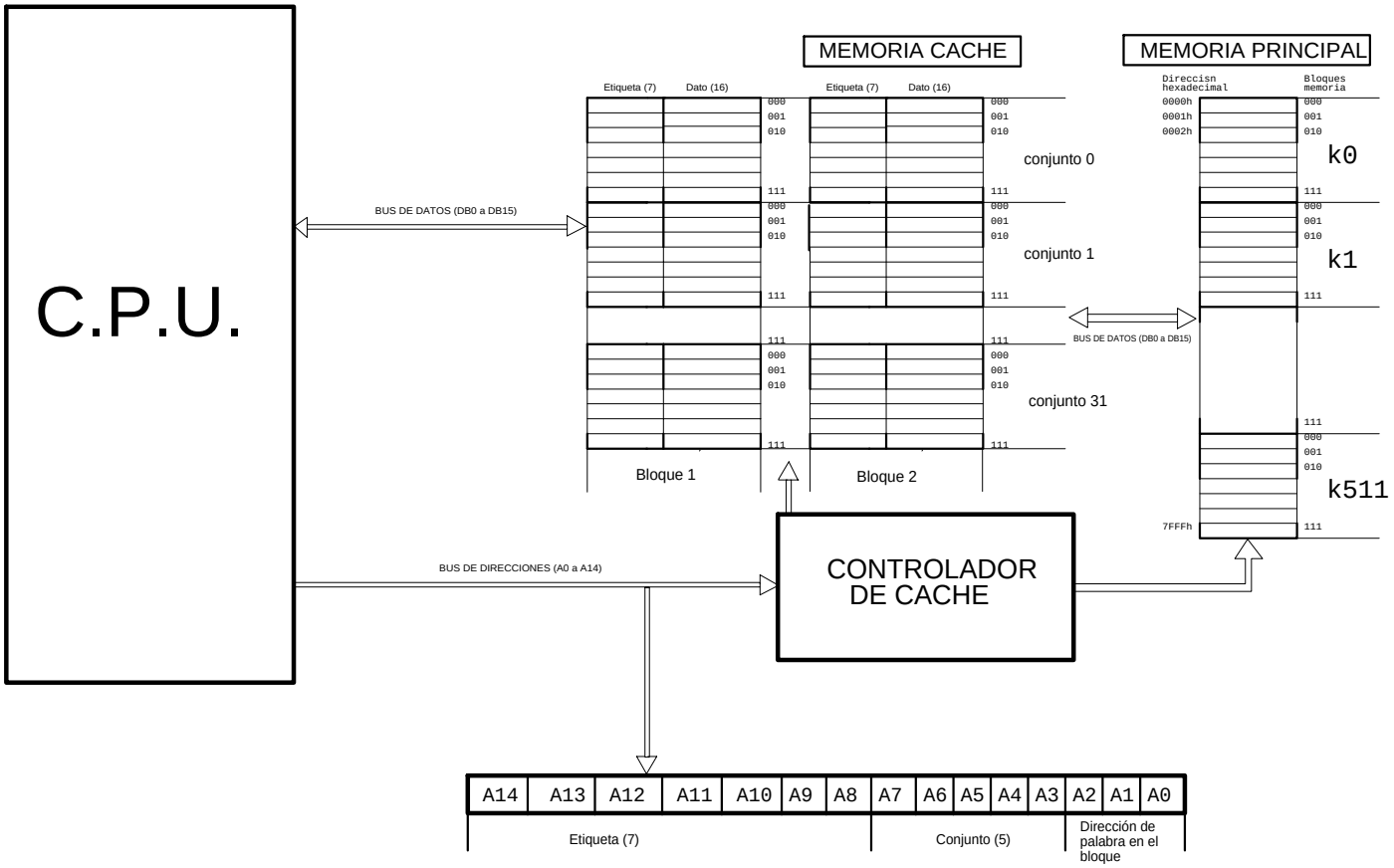
Este proceso se repetirá hasta ocupar los 64 bloques, momento en el cual el próximo bloque antes de entrar deberá liberar uno ya existente, según el algoritmo de reemplazamiento utilizado por el controlador de la memoria caché.

CORRESPONDENCIA ASOCIATIVA POR CONJUNTOS

- Auna las ventajas de los dos métodos anteriores.
- Está compuesta por “r” bloques y “q” conjuntos de modo que C = q×r , siendo C el nº de bloques de la mem. caché.
- El funcionamiento consiste en que cada bloque de la mem. ppal. tiene asignado un conjunto de la caché, pero se puede ubicar en cualquiera de los bloques que pertenecen a dicho conjunto. Ello permite mayor flexibilidad que la correspondencia directa y menor cantidad de comparaciones que la totalmente asociativa.



Esquema de la caché



Funcionamiento:

Principios:

- Palabras por bloque 8 $\Rightarrow$ 3 bits (b0 a b2)
- Nº conjuntos =2 $\Rightarrow$ Nº bloques 64 /2 = 32 $\Rightarrow$ 2^5 $\Rightarrow$ 5 bits
- Etiqueta = (bus direcc) – (bits palab/bloque) -- (bits conjuntos) $\Rightarrow$ 15 –3 – 5 = 7
- Ancho de la mem. caché = Ancho palabra + ancho etiqueta $\Rightarrow$ 16 + 7 = 23 bits

1. La CPU entrega la dirección de mem. al controlador de caché.
2. El ctrl. de caché aísla de los bits b3 a b7 y con ellos apunta al conjunto correspondiente.
3. El ctrl. de caché busca en todos y cada uno de los bloques del conjunto coincidencia entre los bits b14 a b8 del bus de direcciones con la etiqueta.
   - 3.a Si iguales $\Rightarrow$ acierto, con lo que coge de los bits b15 a b0 (dato) de la caché y los saca al bus de datos de la CPU.
   - 4.a Fin de acceso.

   - 4.b Si no encuentra coincidencia $\Rightarrow$ Fallo
   - 5.b El ctrl. de caché transmite bloque completo desde mem. ppal. a mem. caché
     - 5.b.1 El ctrl. Caché busca un bloque dentro del conjunto apuntado de caché para librarlo (algoritmo de reemplazamiento).
     - 5.b.2 Una vez liberado el ctrl. Caché pone a 000 los bits de dir. de palabra (b2 a b0).
     - 5.b.2 El ctrl. Caché apunta al bloque liberado.
     - 5.b.3 El ctrl. Caché lleva de la dir. b14 b13 b12 .....b3 0 0 0 de mem. ppal. a la caché al bloque liberado en el conjunto apuntado por los bits b7 a b3 la dir. de palabra del bloque 0 0 0 (palabra 000 del bloque). Incrementa un contador y lleva de la dir. b14 b13 b12 .....b3 0 0 1 de mem. ppal. a la caché al bloque liberado en el conjunto apuntado por los bits b7 a b3 en la dir. de palabra del bloque 0 0 1 (palabra 001 del bloque). Así sucesivamente hasta la palabra b14 b13 b12 .....b3 1 1 1 (palabra 7 del bloque) . Escribiendo en la zona de la etiqueta el valor de los bits b14 a b8.
     - 5.b.4 Una vez ha cargado el bloque completo (8 palabras) $\Rightarrow$ Hay acierto y se salta al paso 3.a.

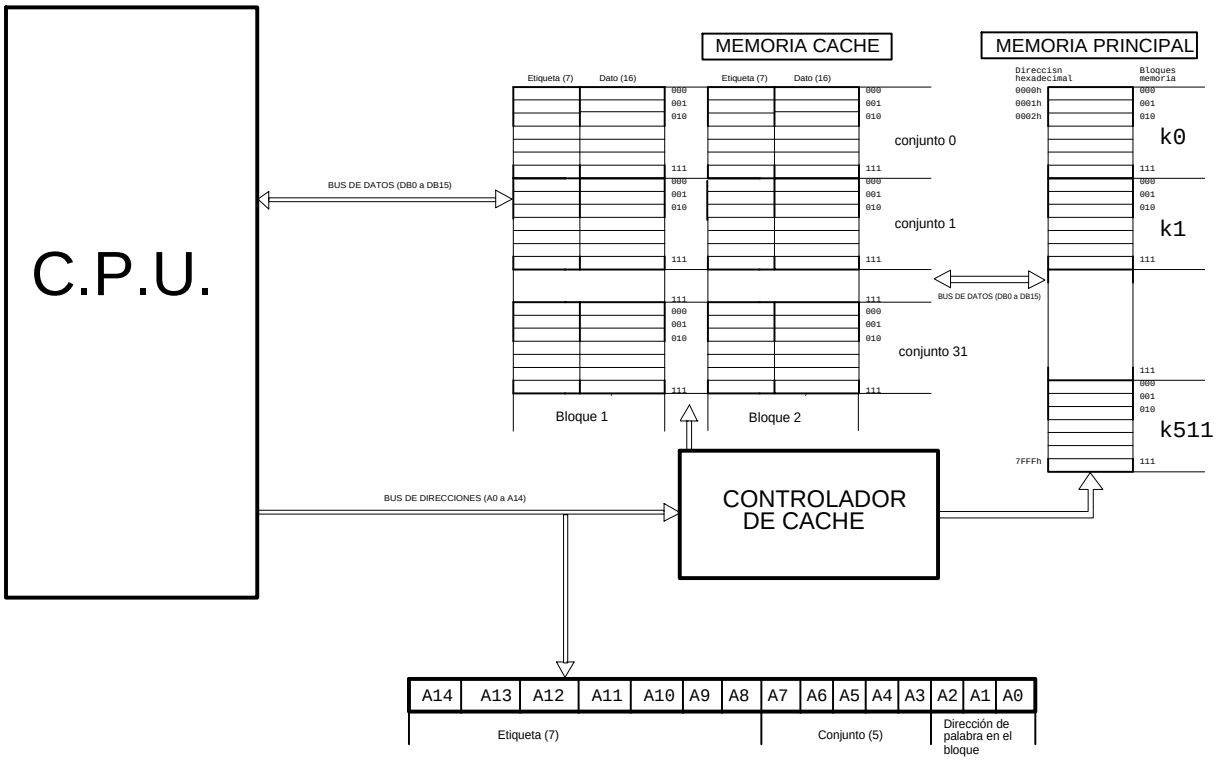
Ejemplo con los parámetros indicados

- Se supone que todavía no ha habido ningún acceso ⇒ Mem. caché vacía ⇒ todo con 0
- Se quiere acceder al contenido de la posición mem. $025F_H$

$025F_H$ =

b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	1	0	0	1	0	1	1	1	1	1
etiqueta							**conjunto**					**Dir. palabra**		

Bits dir. palabra = 3	Bits etiqueta = 7	Bits de conjunto = 5	Ancho palabra = 16



➢ El ctrl. de la caché mira si en el conjunto :

0	1	0	1	1

Hay una etiqueta que contenga

0	0	0	0	0	1	0

Como es el primer acceso la mem. caché contiene todo 0 ⇒ hay primer fallo.

➢ El ctrl. de caché va a direccionar a la mem. ppal. a la dirección

0	0	0	0	0	1	0	0	1	0	1	1	0	0	0

y llevará el contenido al bloque 1 del conjunto seleccionado

0	1	0	1	1

dirección

0	0	0

escribiendo la etiqueta

0	0	0	0	0	1	0

Repitiendo el proceso hasta:

- El ctrl. de caché va a direccionar a la mem. ppal. a la dirección

0	0	0	0	0	1	0	0	1	0	1	1	1	1	1

y llevará el contenido al bloque 1 del conjunto seleccionado

0	1	0	1	1

dirección

1	1	1

escribiendo la etiqueta

0	0	0	0	0	1	0

- Con lo que el bloque completo estará escrito en la mem. caché , existiendo ahora acierto y presentando en el bus de datos el contenido de la mem. caché correspondiente al bloque 1 del conjunto

0	1	0	1	1

Y la dirección de palabra

1	1	1

- Si posteriormente se quiere acceder al contenido de la dirección de mem. $085B_H$

$085B_H$

b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	1	0	0	0	0	1	0	1	1	0	1	1
etiqueta							**conjunto**					**Dir. palabra**		

- El ctrl. de la caché mira si en el conjunto :

0	1	0	1	1

Hay una etiqueta que contenga

0	0	0	0	0	1	0

Como no existe porque todavía no ha sido cargada $\Rightarrow$ hay fallo.

- El ctrl. de caché va a direccionar a la mem. ppal. a la dirección

0	0	0	1	0	0	0	0	1	0	1	1	0	1	1

y llevará el contenido al bloque 2 del conjunto seleccionado

0	1	0	1	1

dirección

0	0	0

escribiendo la etiqueta

0	0	0	1	0	0	0

Repitiendo el proceso hasta:

- El ctrl. de caché va a direccionar a la mem. ppal. a la dirección

0	0	0	1	0	0	0	0	1	0	1	1	1	1	1

y llevará el contenido al bloque 1 del conjunto seleccionado

0	1	0	1	1

dirección

1	1	1

escribiendo la etiqueta

0	0	0	1	0	0	0

- Con lo que el bloque completo estará escrito en la mem. caché , existiendo ahora acierto y presentando en el bus de datos el contenido de la mem. caché correspondiente al bloque 1 del conjunto

0	1	0	1	1

Y la dirección de palabra

1	1	1

- Si posteriormente se quiere acceder al contenido de la dirección de mem. $005B_H$

$005B_H$

b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	0	0	1	0	1	1	0	1	1
etiqueta							**conjunto**					**Dir. palabra**		

- El ctrl. de la caché mira si en el conjunto :

0	1	0	1	1

Hay una etiqueta que contenga

0	0	0	0	0	0	0

Como no existe $\Rightarrow$ hay fallo.

- El ctrl. de caché liberará uno de los dos bloques del mencionado conjunto según algoritmo de reemplazamiento y procederá según los pasos indicados anteriormente al cargar el nuevo bloque.

- Algoritmos de reemplazamiento

Decisión de bloque a eliminar de la mem. caché cuando hay que dejar sitio a un nuevo bloque:

- Directa ⇒ No hay algoritmo porque cada bloque tiene su sitio
- Totalmente asociativa ⇒ elegir entre todos los bloques
- Asociativa por conjuntos ⇒ " " bloques de un conjunto

- Algoritmos
  - Bloque elegido aleatoriamente → {-Fácil implementar, -Rápido}
  - Menos utilizado frecuentemente (L.F.U.)
  - Más antigüo en caché (FIFO) (First Input First Output)
  - Menos utilizado recientemente (LRU)

- Estrategia de escritura

↳ Modo de actualizar la mem. principal con los valores escritos en la caché por la CPU

Problemas
- Más de un dispositivo con acceso a Mem. ppal
- Varias CPU al mismo bus y cada una su caché

Técnicas
- Escritura directa → siempre que se escribe en la caché se actualiza la principal
  Tráfico alto → cuello de botella
- Postescritura → se activa un bit de actualización de forma que se pone a "1" cuando hay cambio en la caché y que avisa para postescritura

- Tamaño del bloque

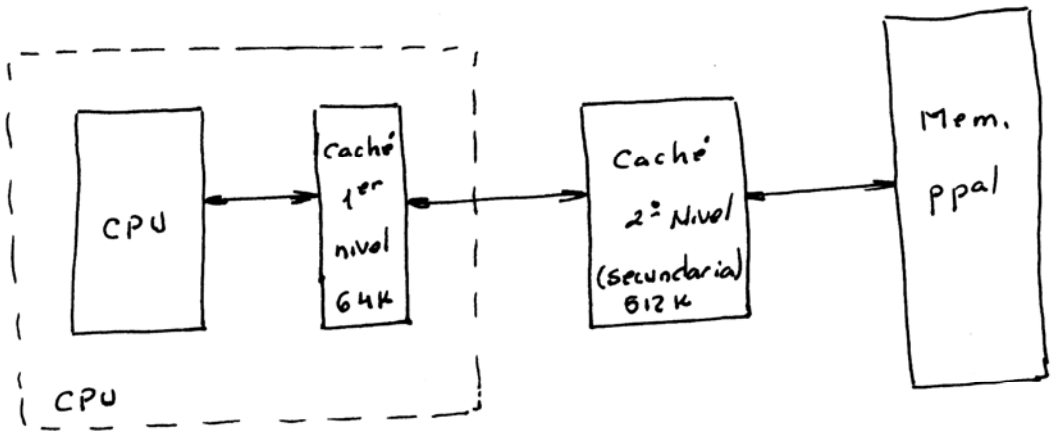
Compromisos
- Mayor tamaño bloque → menor nº de ellos
- Mayor tamaño → cada palabra añadida está a mayor distancia de la solicitada por la CPU ⇒ Pierde principio de localidad

Tamaño bloque → relación compleja → Tasa acierto

Tamaño razonable 4/8 palabras

- Número de cachés

Solución para evitar aumentar el tamaño de la caché es poner dos niveles



CPU → 1º → 1er Nivel → si no está → 2º Nivel → si está →

→ bloque a 1er Nivel y a CPU.

⇓

Todos los bloques del 1er Nivel están en el 2º Nivel

⇓

Principio de inclusión

Algoritmo de reemplazamiento → LRU
Estrategia de escritura → directa

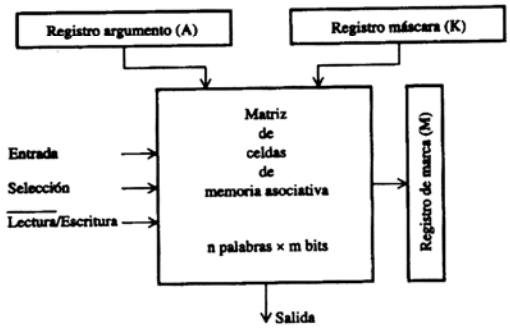
5.- Memorias asociativas (Mem. direccionables por contenido) CAM

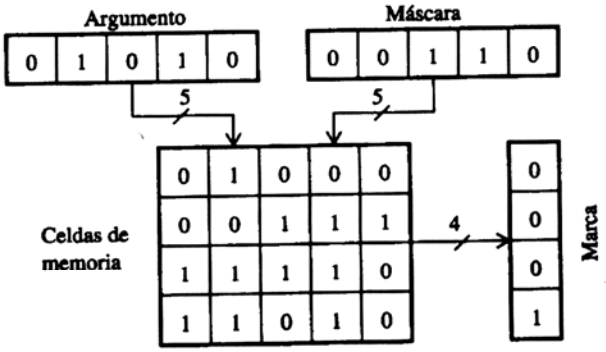
Acceso → especificando su contenido o parte de él

Concepto → En la mem en todas las direcciones hay un campo para la referencia de búsqueda y otro campo con el dato

Ejemplo → Mem. caché totalmente asociativa
↓
Elemento de búsqueda la etiqueta

Estructura
- Matriz de celdas de memoria
- Reg. argumento (A) → patrón
- " máscara (K) → aisla bits del patrón
- " de marca (M) → indica coincidencias





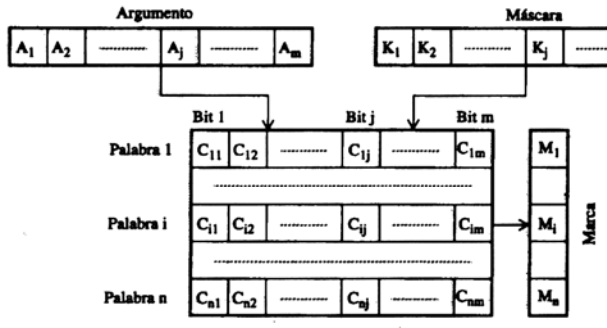
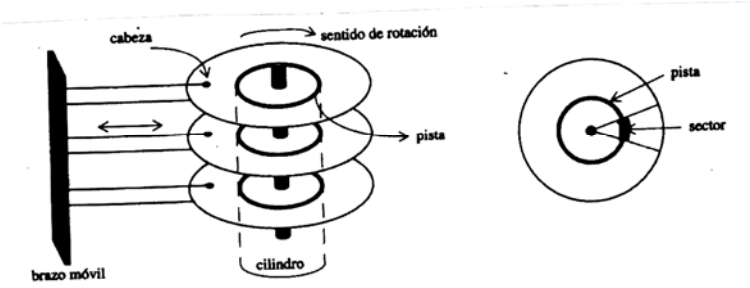


Figura 2.51: Memoria asociativa de n palabras × m bits

8.- Discos magnéticos

Estructura física

- Película de óxido magnético sobre soporte inerte (aluminio o plástico)
  Plato
- Platos { - Extraibles
  - Fijos



Los discos giran a velocidad cte y varían de 1 a 20. discos.

Estructura
- Cabezas lectu/escrit → 1 por cara
- Cada disco 2 superficies
- Pistas → concentricas
- Cilindro → conjunto de pistas paralelas de todas superfi.
- Sector → Porción continuada de bits en que se divide cada pista

Velocidad giro cte
Nº sect / pista cte en todas pistas
Nº bytes / secto cte
Pistas diferente radio
⇒ Densidad de grabación diferente en las diferentes pistas

Tiempos

Tipo acceso = directo ⇒ { -aleatorio → posicionar cabeza + -secuencial → transferir datos

- Tiempo de búsqueda (t_b) = tiempo para posicionar la cabeza en el cilindro correspondiente

t_b = t_i (arranque inicial) + t. recorrer los cilindros

$t_b = m \times n + t_i$ { m = cte del disco; n = nº cilindro a desplazar

- Tiempo de latencia rotacional (retardo rotacional) (t_r) = tiempo que se tarda en girar el disco y posicionar el sector.

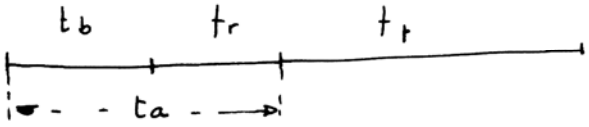
Se toma el valor medio = máximo/2

$t_r = \frac{1}{2f}$ { f = velocidad de rotación

- Tiempo de acceso = $t_b + t_r$

- Tiempo de transferencia (t_t) = tiempo en transferir los datos una vez posicionada la cabeza

$t_t = \frac{b}{p \cdot f}$ { b = nº bytes a transferir; p = nº bytes / pista



Ejemplo

Suponer la lectura de un archivo de 256 KB todo consecutivo.

$t_b = 25$ mseg

vel. transfer = 800 KB/s

sector = 256 B

sectores/pista = 64

veloc. giro = 3600 rpm $\Rightarrow$ $t_{vuelta} = 16.6$ mseg

y

$t_r = 8.3$ mseg

N° sectores = $\frac{256\,KB}{} \Rightarrow$ 1024 sectores

N° pistas = $\frac{N°\,sect}{sec/pista} = \frac{1024}{64} = 16$ pistas

t_t/pista = $\frac{256 \cdot 64}{800} = 20,48$ mseg

Secuencial

1ª pista $\begin{cases} t_b = 25 \text{ mseg} \\ t_r = 8,3 \text{ mseg} \\ t_t = 20,48 \text{ mseg} \end{cases} \Rightarrow t = 63,78$ mseg

resto (15) $\begin{cases} t_r = 8,3 \text{ mseg} \\ t_t = 20,48 \text{ mseg} \end{cases} \Rightarrow t = 28,78$ mseg

$t_{total} = 63,78 + (15 \cdot 28,78) = \underline{485,48 \text{ mseg}}$

Aleatorio

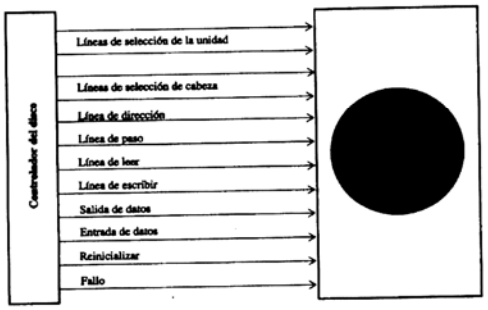
$t_b = 25$ mseg

$t_r = 8,3$ mseg

t_{sector} $\Rightarrow$ sector = 256 B $\rightarrow$ 800 KB/seg $\Rightarrow$ 0,32 mseg

$t_{total} = n° sec \times (t_{sec} + t_b + t_r) 1024 \times (25 + 0,32 + 8,3) = \underline{34427 \text{ mseg}}$

Controlador de disco

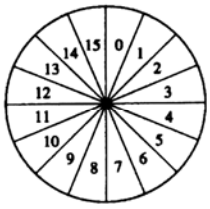


Funciones
- Control errores
  - Transitorio ⇒ relectura
  - Permanente ⇒ marcado defectuoso
- Buffer interno de almacenamiento temporal

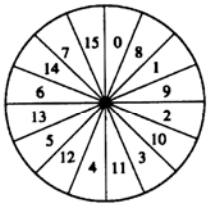
Funcionamiento: Lee sector, analiza y si error relectura y si no error continua ⇒ No puede leer sectores consecutivos ; ya que para leer 2 sect. consecutivos ha de dar 2 vueltas ⇒ bajo rendimiento.

⇓

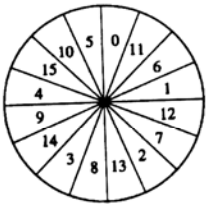
Mejora ⇒ entrelazado



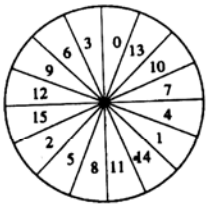
sin entrelazado



entrelazado simple



entrelazado doble



entrelazado cuadruple

Direccionamiento del disco
- Lógico → Nº consecutivo del 0 a n sectores
- Físico ⇒ <cilindro, cabeza, sector>

Planificación del disco

Información de una operación de E/S del disco:
- Tipo operación
- Dirección en el disco
- Dirección en memoria
- Nº bytes a transferir

Buffer intermedio → almacena los sectores temporalmente para luego acceder al disco.

Planificaciones de acceso a los sectores:
- FCFS (First Come First Served) ⇒ FIFO ⇒ 1º entrar 1º salir
- SSTF (Shortest service time first) ⇒ 1º el más cercano
- SCAN (rastreo) → todas pistas en una dirección u otra
- C-SCAN → única dirección
- LOOK / C-LOOK ⇒ = SCAN pero sin llegar al fin

Ejemplo suposición ⇒ 22, 124, 105, 181, 142, 36, 5, 59, 115

Próxima pista a la que se accede	22	124	105	181	142	36	5	59	115	
Número de pistas que se atraviesan	73	102	19	76	39	106	31	54	56	LMB = 61,8

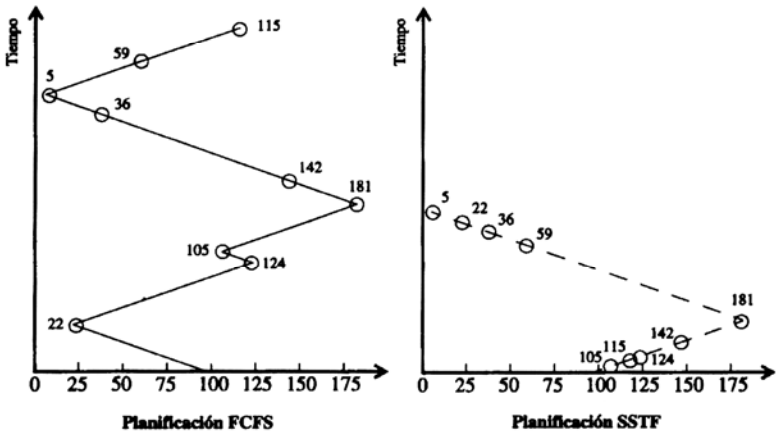
Tabla 2.18: Algoritmo FCFS de planificación del disco

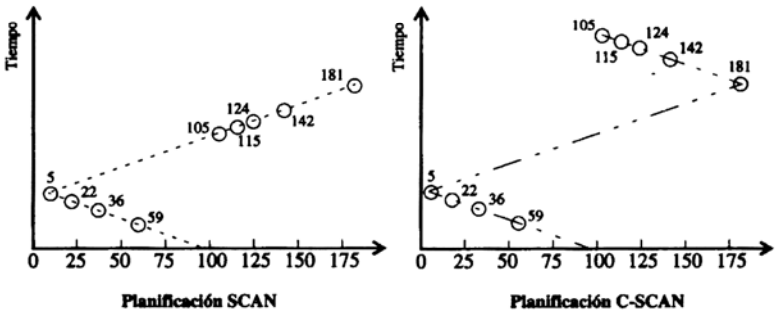
na pista a la que ede	105	115	124	142	181	59	36	22	5	
ro de pistas que aviesan	10	10	9	18	39	122	23	14	17	LMB = 29,1

Tabla 2.19: Algoritmo SSTF de planificación del disco

Próxima pista a la que se accede	59	36	22	5	105	115	124	142	181	
Número de pistas que se atraviesan	36	23	14	17	100	10	9	18	39	LMB = 29,5

Tabla 2.20: Algoritmo SCAN de planificación del disco





Longitud media de búsqueda: $\frac{\Sigma \text{ de pistas recorridas}}{\text{Nº de pistas recorridas}}$