



## TEMA 6: ORGANIZACIÓN DE MEMORIA

- 1. Introducción**
- 2. Jerarquía de Memoria**
- 3. Memorias de Semiconductores**
- 4. Configuración y Diseño de Memorias**
- 5. Memorias Asociativas**
- 6. Memorias Caché**



## ➔ Introducción

1. Objetivos.
2. Clasificación.
3. Modo de Acceso.
4. Métodos para Aumentar el Ancho de Banda.



Las prestaciones que puede ofrecer un ordenador dependen en gran medida de:

- La **CAPACIDAD** de almacenamiento de memoria.
- La **VELOCIDAD** de acceso a memoria.
- La **ORGANIZACIÓN** de la memoria.

Según esto algunos **OBJETIVOS** a tener en cuenta en el diseño del sistema de memoria son:

- **CAPACIDAD** de almacenamiento y **VELOCIDAD** de acceso suficientes.
- **COSTE** por bit reducido.
- Los **PROGRAMADORES** deben estar **LIBRES** de realizar tareas de **GESTIÓN** de memoria.



Aquí consideramos la memoria en un sentido amplio, englobando:

- **MEMORIA INTERNA:** REGISTROS. No hay diferencia de velocidad con la CPU.
- **MEMORIA CACHÉ.**
- **MEMORIA PRINCIPAL:** La CPU sólo puede acceder a programas y datos que estén en la memoria principal.
- **MEMORIA SECUNDARIA o MASIVA:** Más lenta. DISCOS y CINTAS magnéticas.

**Consideraremos TIEMPO DE ACCESO como el tiempo que se requiere para leer/escribir un dato (palabra) desde/en la memoria.**



**Según el MODO DE ACCESO clasificamos las memorias en tres clases:**

- De **ACCESO ALEATORIO** (RAM). El tiempo de acceso es independiente de la posición de memoria a acceder. Ejs: RAM, ROM.
- De **ACCESO SECUENCIAL** (SAM). El tiempo de acceso depende de la posición de los datos. Ej: Cinta magnética.
- De **ACCESO SEMIALEATORIO**
  - **DIRECTO** (DASD: “Direct Access Storage Device”). Principalmente discos, en los que se accede a las pistas de forma casi aleatoria y a los datos dentro de las pistas de forma secuencial.

**Hay que tener en cuenta que en SAM y DASD el tiempo de acceso a un bloque de  $n$  palabras NO es  $n \times \text{Tiempo de acceso}$ , sino:**

**Tiempo de acceso + Tiempo de bloque**

(en acceder a la  
primera palabra)

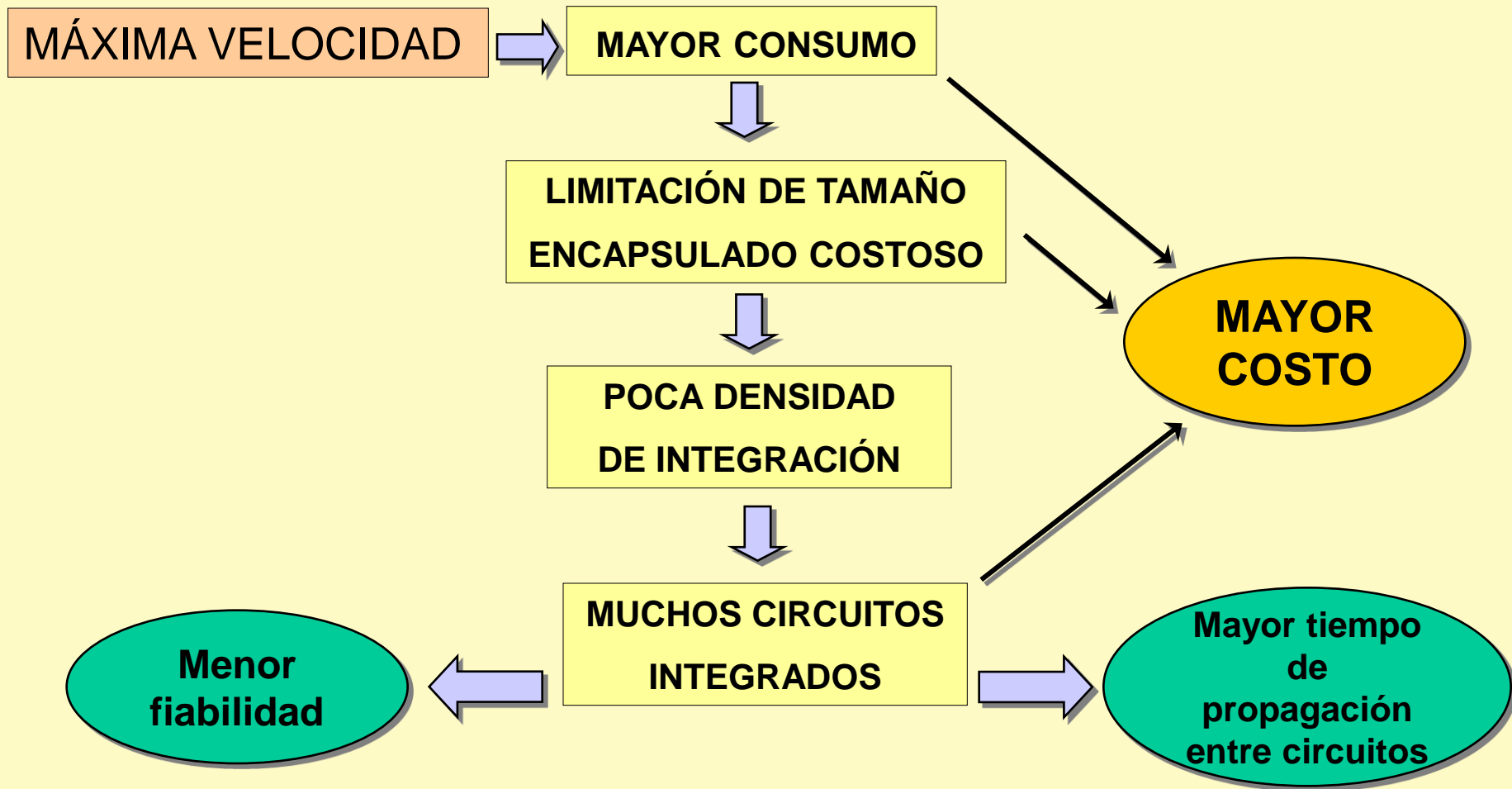


## MÉTODOS PARA AUMENTAR EL ANCHO DE BANDA DE MEMORIA

1. Usar tecnología de ALTA VELOCIDAD
2. Organizar la memoria JERÁRQUICAMENTE, incluyendo MEMORIA CACHÉ
3. Incrementar el ANCHO DE LA MEMORIA (Número de palabras accedidas simultáneamente)
4. Dividir la memoria principal en un conjunto de módulos direccionables simultáneamente (MEMORIA ENTRELAZADA)
5. Emplear MEMORIA ASOCIATIVAS (Direcciones por contenido) cuando sea necesario



Disponer de memoria de máxima capacidad con el menor tiempo de acceso es inviable.





- 1. Introducción
- 2. Jerarquía de Memoria.
- 3. Memorias de Semiconductores
- 4. Configuración y Diseño de Memorias
- 5. Memorias Asociativas
- 6. Memorias Caché

## TEMA 6: ORGANIZACIÓN DE MEMORIA

1. Introducción

➔ **2. Jerarquía de Memoria**

3. Memorias de Semiconductores

4. Configuración y Diseño de Memorias

5. Memorias Asociativas

6. Memorias Caché





## → Jerarquía de Memoria

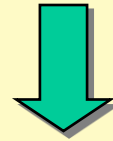
1. Regla 90/10.
2. Localidad de las Referencias.
3. Jerarquía de Niveles de Memoria.
4. Parámetros de los Dispositivos de Almacenamiento.
5. Propiedad de Inclusión y Coherencia de Datos.
6. Modelo de Evaluación del Rendimiento.
7. Problemas a Resolver en Cualquier Nivel.



## JERARQUÍA DE MEMORIA

Aprovecha los siguientes hechos:

1. **REGLA 90/10:** Una regla empírica ampliamente corroborada es que un programa pasa el 90% de su tiempo de ejecución en sólo el 10% de su código.



**Basándonos en el pasado reciente de un programa, podemos predecir con una predicción razonable qué instrucciones y datos utilizará en un futuro próximo.**



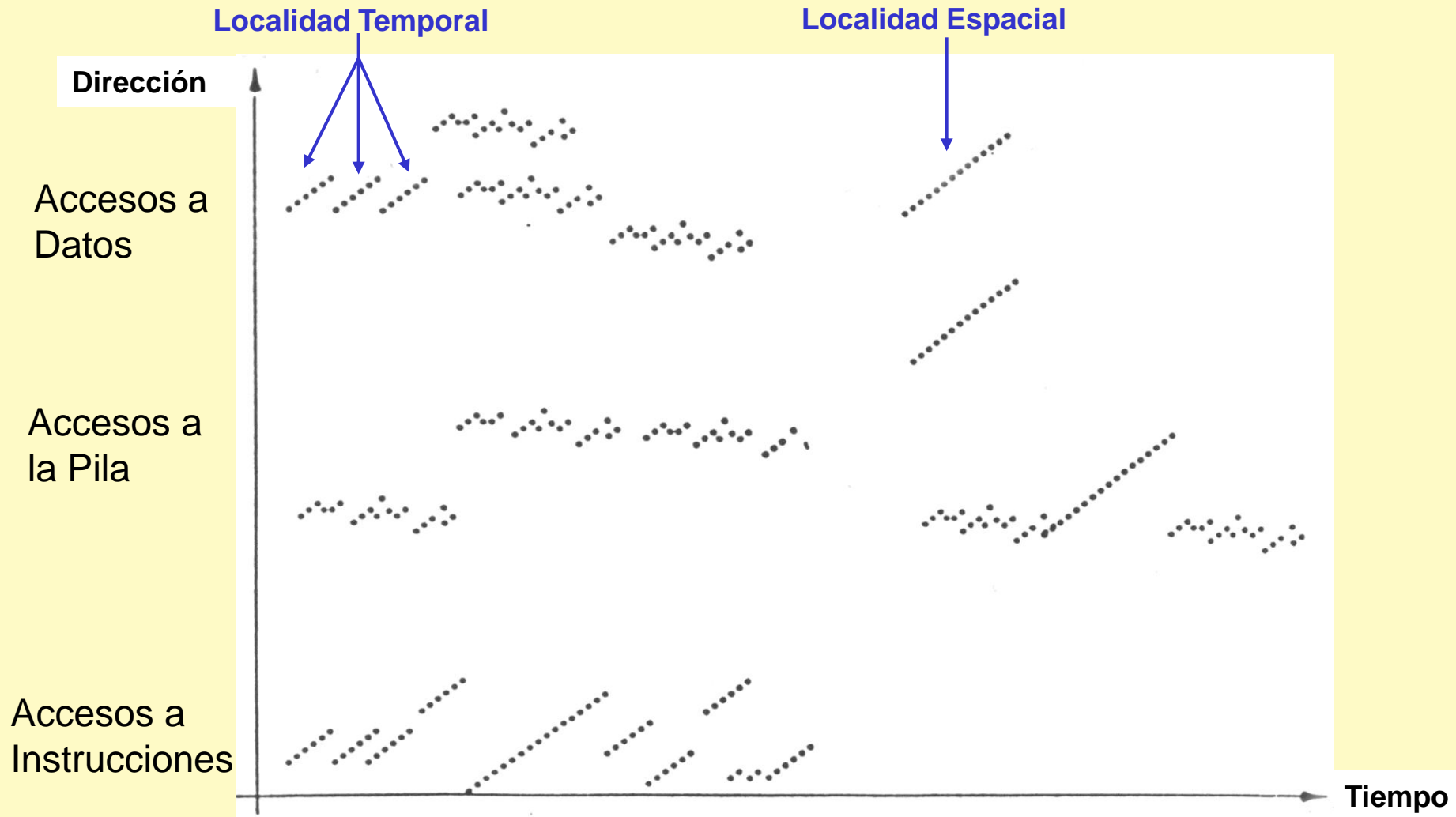
Esta localidad tiene 2 dimensiones:

LOCALIDAD DE LAS REFERENCIAS

- **LOCALIDAD ESPACIAL**  $d(t) + k = d(t+n)$  con  $n$  y  $k$  pequeños  
“Si se referencia un elemento, los elementos cercanos a él tenderán a ser referenciados pronto”.  
“Las direcciones de memoria que se están utilizando suelen ser contiguas”.  
**JUSTIFICACIÓN:** Los datos relacionados se almacenan juntos.  
Las instrucciones se ejecutan secuencialmente.  $k = n = 1$
- **LOCALIDAD TEMPORAL**  $d(t) = d(\tau+n)$  con  $n$  pequeño  
“Si se referencia un elemento, tenderá a ser referenciados pronto”.  
“La información que se usará en un futuro próximo es aproximadamente la misma que se está usando actualmente”.  
**JUSTIFICACIÓN:** Bucles.  
Uso de datos de forma repetitiva.  
Llamadas repetidas a subrutinas.



Patrones de referencia a memoria típicos:

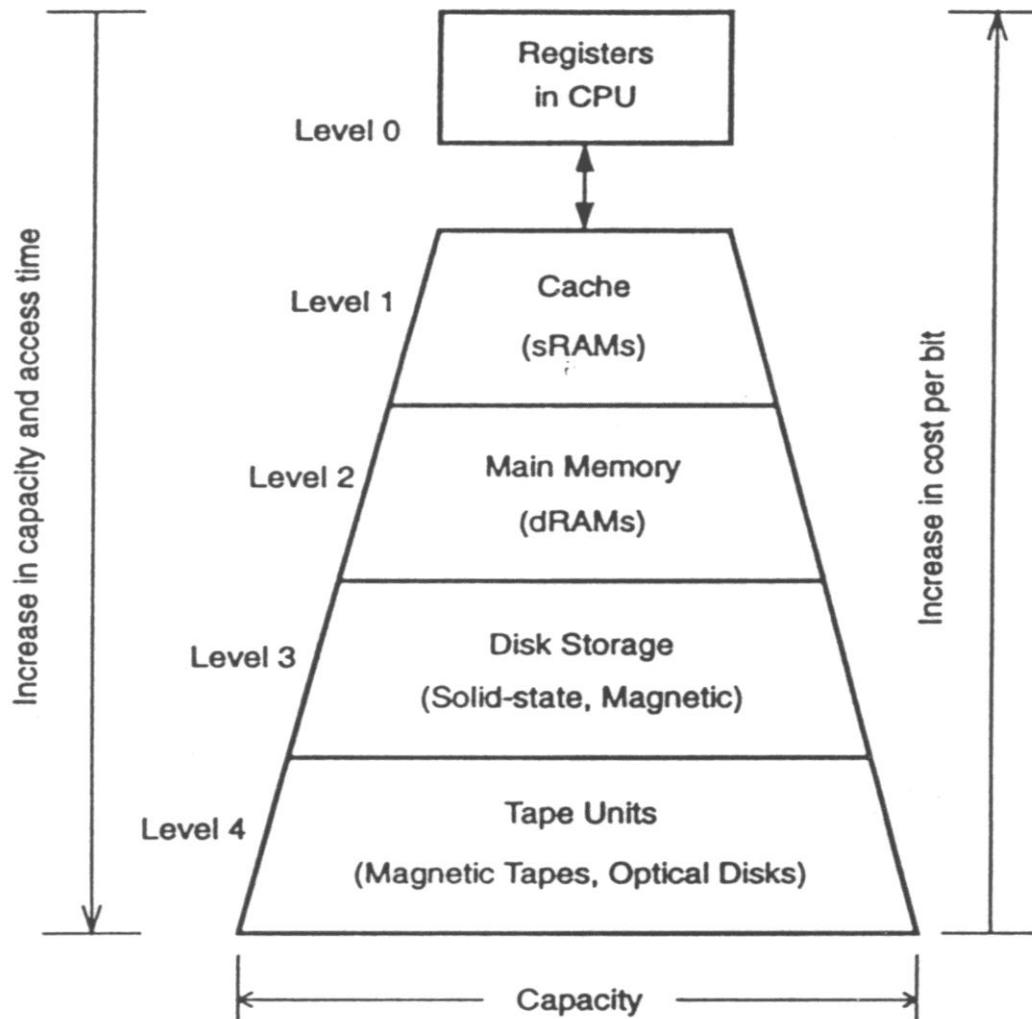




Teniendo en cuenta las limitaciones tecnológicas, y aprovechando el principio de localidad se puede conseguir un sistema de memoria eficaz mediante una

## **JERARQUÍA DE NIVELES DE MEMORIA:**

- ❖ Cada nivel tiene un CAPACIDAD MENOR pero es MÁS RÁPIDO que los inferiores.
- ❖ Toda la INFORMACIÓN de un nivel se encuentra ALMACENADA también en el SIGUIENTE.



← Gestionados por el compilador (asignación de registros) y el hardware (procesador)

← Gestionada por el hardware (unidad de gestión de memoria o MMU)

← Gestionada por el hardware (MMU) y por el SO

← Gestionados por el SO con una intervención limitada del usuario.



Los dispositivos de almacenamiento, como los registros, las memorias, los discos y las unidades de cinta, se caracterizan por cinco parámetros:

- **Tiempo de Acceso** ( $t_i$ ) : Tiempo desde que se inicia una lectura hasta que llega la palabra deseada.
- **Tamaño de la Memoria** ( $s_i$ ) : Número de bytes, palabras, sectores, etc. que se pueden almacenar en el dispositivo de memoria.
- **Coste por Byte** ( $c_i$ ).
- **Ancho de Banda** ( $b_i$ ) : Velocidad a la que se transfiere información desde un dispositivo.
- **Unidad de Transferencia** ( $x_i$ ) : Tamaño de la unidad de la información que se transfiere entre el nivel  $i$  y el  $i+1$ .

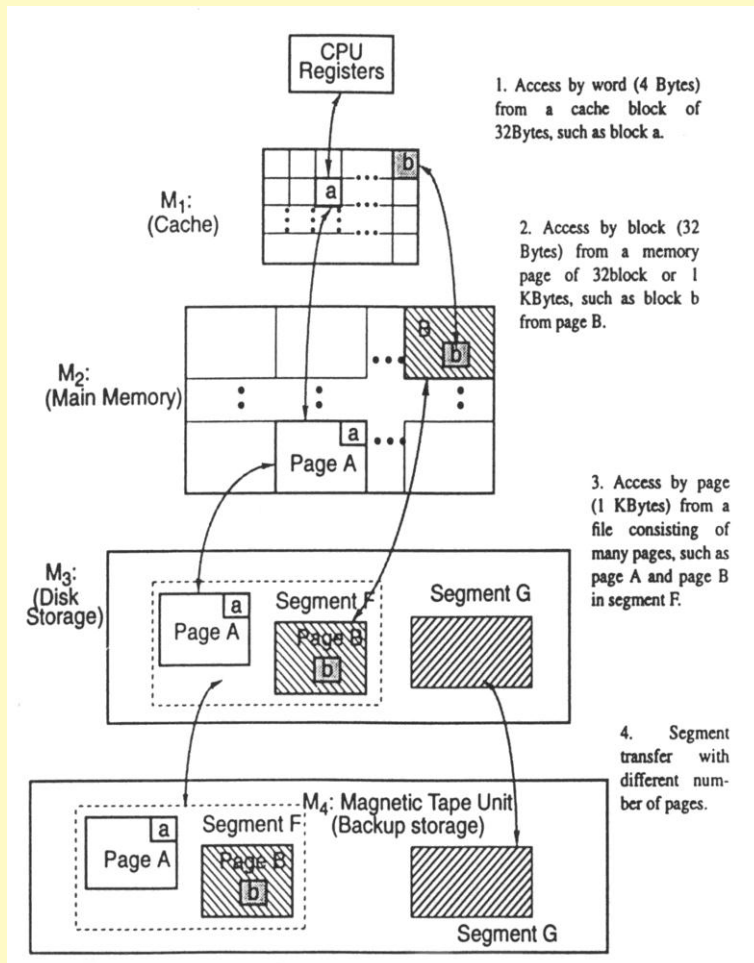
Se verifica que:

$$\begin{array}{ll} t_i < t_{i+1} & c_i > c_{i+1} \\ s_i < s_{i+1} & b_i > b_{i+1} \\ x_i < x_{i+1} & \end{array}$$



## 2. PROPIEDAD DE INCLUSIÓN Y COHERENCIA DE DATOS

Propiedad de Inclusión:  $M_1 \subset M_2 \subset M_3 \subset \dots \subset M_n$



Si una palabra se encuentra en  $M_i$ , copias de esa palabra también se encuentran en  $M_{i+1}$ ,  $M_{i+2}$ , ...,  $M_n$ .

Sin embargo, una palabra almacenada en  $M_{i+1}$  puede no estar en  $M_i$ , y si es así, tampoco estará en

$M_{i-1}$ ,  $M_{i-2}$ , ...,  $M_1$ .





**Penalización en el tiempo de acceso**

LECTURA: No toda la información que necesita la CPU está en  $M_1$ . Si la palabra deseada no se encuentra en el nivel 1 se intentará localizar en los niveles inferiores. Supóngase que está en  $M_j$ ,  $j > 1$ . Se transferiría a  $M_{j-1}$ , luego a  $M_{j-2}$ , y así sucesivamente hasta llegar al nivel 1, aunque puede haber “atajos”. Para ahorrar tiempo y aumentar la eficiencia de las transferencias, y aprovechando el principio de localidad, **se transfieren bloques completos** en lugar de transferir sólo la palabra requerida.

ESCRITURA: Se da un problema de **CONSISTENCIA o COHERENCIA DE DATOS** entre los distintos niveles. Si una palabra se modifica en un nivel superior, más tarde o más temprano tiene que escribirse en niveles inferiores. En general hay dos ESTRATEGIAS PARA MANTENER LA COHERENCIA:

- Write – Through (Escritura Directa): Si se modifica una palabra en  $M_i$ , se modifica inmediatamente en  $M_{i+1}$ .
- Write – Back (Post-Escritura): Se retrasa la actualización en  $M_{i+1}$  hasta que la palabra modificada en  $M_i$  sea reemplazada o borrada de  $M_i$ .



## MODELO DE EVALUACIÓN DEL RENDIMIENTO DE UNA JERARQUÍA DE MEMORIA

(C. K. Chow, 1974)

Se suele considerar que la política de administración de la memoria viene caracterizada por una función de éxito o tasa de aciertos  $A$ .

**TASA DE ACIERTOS  $A_i$**  : Probabilidad de que la información que se busca se encuentre en el nivel  $i$ .  
(HIT RATIO)

En una jerarquía con  $n$  niveles:  $A_0 = 0$   $A_1 = 1$

$A_i$  depende de:

- la capacidad del nivel  $i$  ( $S_i$ ).
- la granularidad de la transferencia de información.
- la estrategia de administración de memoria.

**TASA DE FALLOS  $F_i$**  :  $F_i = 1 - A_i$ ,  $i = 0, \dots, n$   $F_0 = 1$   $F_n = 0$   
(MISS RATIO)



**$a_i$**  : Probabilidad de acceder con éxito a una información en el nivel  $i$  y que esa información no se encuentre en los niveles 0 a  $i-1$  .

Como la información de  $M_j$  está también en  $M_k$  , con  $k > j$ :

$$a_i : A_i - A_{i-1} , \quad i = 1, \dots, n \quad a_1 = A_1 \quad a_n = 1 - A_{n-1}$$

También se puede ver como la frecuencia de accesos con éxito al nivel  $i$  :

$$\sum_{i=1}^n a_i = 1$$

Los OBJETIVOS al diseñar una memoria con  $n$  niveles son:

- 1.- Obtener un **rendimiento** cercano al de la memoria  $M_1$  (la más rápida)
- 2.- Obtener un **coste por bit** cercano al de la memoria  $M_n$  (la más barata)



1. El rendimiento de una jerarquía se puede medir por el **TIEMPO MEDIO DE ACCESO DE LA JERARQUÍA PARA CADA REFERENCIA A MEMORIA** ( $\bar{T}$ )

Tiempo de acceso efectivo del procesador al nivel  $i$ -ésimo de la jerarquía:

$$T_i = \sum_{j=1}^i t_j \quad (t_j \approx \text{tiempo de acceso medio individual de cada nivel})$$

$$\bar{T} = \sum_{i=1}^n a_i \cdot T_i \quad (\text{Sumatoria de: la frecuencia de acceso con éxito a } M_i \text{ multiplicada por el tiempo de efectivo de acceso a } M_i)$$

Sustituyendo  $a_i$  y  $T_i$ :

$$\begin{aligned} \bar{T} &= \sum_{i=1}^n \left[ (A_i - A_{i-1}) \cdot \sum_{j=1}^i t_j \right] = (A_1 - A_0) t_1 + (A_2 - A_1) (t_1 + t_2) + (A_3 - A_2) (t_1 + t_2 + t_3) + \\ &\quad + \dots + (A_n - A_{n-1}) (t_1 + t_2 + t_3 + \dots + t_n) = \\ &\quad (\cancel{A_1} - A_0 + \cancel{A_2} - \cancel{A_1} + \cancel{A_3} - \cancel{A_2} + \dots + A_n - \cancel{A_{n-1}}) t_1 + \\ &\quad + (\cancel{A_2} - A_1 + \cancel{A_3} - \cancel{A_2} + \dots + A_n - \cancel{A_{n-1}}) t_2 + \\ &\quad + (\cancel{A_3} - A_2 + \dots + A_n - \cancel{A_{n-1}}) t_3 + \\ &\quad + \dots + (A_n - A_{n-1}) t_n = \\ &\quad = \sum_{i=1}^n (A_n - A_{i-1}) t_i = \sum_{i=1}^n (1 - A_{i-1}) t_i = \sum_{i=1}^n F_{i-1} \cdot t_i \end{aligned}$$



2. El **coste por bit promedio**  $C(n)$  de un sistema de  $n$  niveles es:

$$C(n) = \frac{\sum_{i=1}^n C_i S_i}{\sum_{i=1}^n S_i} + C_0$$

Coste de interconexión entre niveles

$$C_{\text{total}} = \sum_{i=1}^n C_i S_i$$



## PROBLEMAS A RESOLVER EN CUALQUIER NIVEL DE LA JERARQUÍA DE MEMORIA

1. COLOCACIÓN O UBICACIÓN DE BLOQUES.  
*¿ Dónde se ubicará un bloque en el nivel  $i$  cuando se transfiere a éste desde el  $i+1$ ?*
2. IDENTIFICACIÓN DE BLOQUES.  
*¿ Cómo se encuentra un bloque si está en el nivel  $i$  ?*
3. SUSTITUCIÓN O REEMPLAZO DE BLOQUES.  
*Si se produce un fallo en el nivel  $i$  y hay que traer un nuevo bloque desde  $M_{i+1}$  , y suponiendo que  $M_i$  está llena, ¿qué bloque del nivel  $i$  hay que reemplazar?*
4. ESTRATEGIA DE ESCRITURA.  
*¿Qué acciones hay que llevar a cabo si una escritura modifica un bloque en el nivel superior?*

Veremos estos problemas en el caso de:

- Memorias Caché:  $i \approx$  Memoria Caché  
 $i+1 \approx$  Memoria Principal
- Memoria Virtual:  $i \approx$  Memoria Principal  
 $i+1 \approx$  Disco



1.Introducción

3.Jerarquía de Memoria

3. Memorias de Semiconductores

4. Configuración y Diseño de Memorias

5. Memorias Asociativas

6. Memorias Caché

## TEMA 6: ORGANIZACIÓN DE MEMORIA

1. Introducción

2. Jerarquía de Memoria

→ **3. Memorias de Semiconductores**

4. Configuración y Diseño de Memorias

5. Memorias Asociativas

6. Memorias Caché



## ➔ Memorias de Semiconductores

1. Clasificación.
2. Celda de Memoria SRAM.
3. Celda de Memoria DRAM.
4. Circuitos de Memoria DRAM.





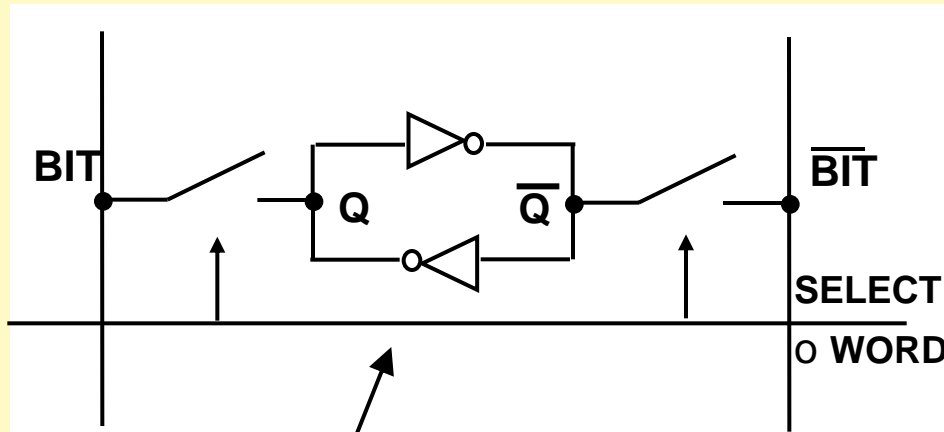
## MEMORIAS DE SEMICONDUCTORES

**Son las empleadas para la memoria principal y la memoria caché. Todas son de acceso aleatorio.**

**CLASIFICACIÓN:**

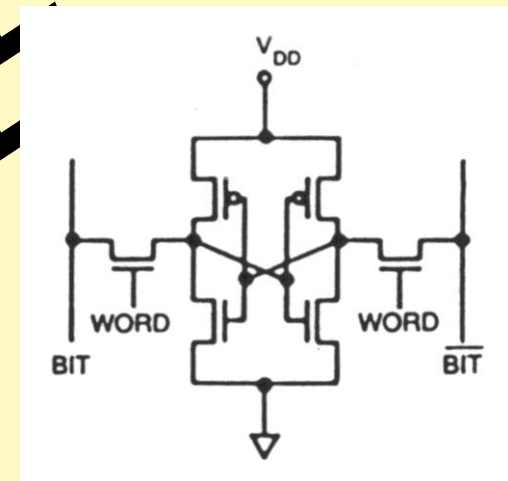
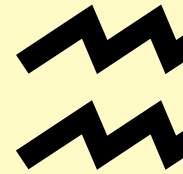
- **DE LECTURA Y ESCRITURA o RAM**
  - Estáticas o SRAM → **Para caché.**
  - Dinámicas o Con Refresco o DRAM.
    - VRAM (Memorias de Vídeo). **Permiten ser accedidas simultáneamente por el procesador y por la circuitería de refresco de vídeo (acceso muy rápido de forma serie).**
- **DE SÓLO LECTURA**
  - ROM (Read Only Memory) → **La información se graba en el proceso de fabricación.**
  - PROM (Programmable Read Only Memory) → **La información se graba en un proceso posterior irreversible.**
  - EPROM ( Erasable Programmable Read Only Memory) → **Se pueden borrar mediante radiación ultravioleta.**
  - EEPROM ( Electrically Erasable Programmable Read Only Memory) → **Se pueden borrar mediante elevadas corrientes.**

## CELDA DE MEMORIA SRAM



Bi-estable formado por dos inversores acoplados

Estática → Los datos almacenados se mantienen por un tiempo indefinido si hay alimentación



**CELDA SRAM CMOS**

- El dato almacenado está representado por  $Q$  y  $\overline{Q}$ .
- Las líneas de bit  $BIT$  y  $\overline{BIT}$  se utilizan para transferir datos desde o hacia la celda.
- Cuando la línea  $SELECT$  o  $WORD$  esté a:
  - “1” se selecciona la celda para lectura o escritura.
  - “0” los transistores no conducen, desconectando la celda de las líneas de bit.



## OPERACIÓN DE LECTURA

- BIT y  $\overline{\text{BIT}}$  se ponen a “1” ‘débil’.
- Se selecciona la celda poniendo SELECT o WORD a “1”, con lo cual Q se conecta a BIT y  $\overline{Q}$  a  $\overline{\text{BIT}}$ .
- Si  $Q=1$  ( $\overline{Q} = 0$ ), la línea  $\overline{\text{BIT}}$  se pone a 0.  
 $\text{BIT} = 1$  y  $\overline{\text{BIT}} = 0 \rightarrow$  se lee un 1
- Si  $Q=0$  ( $\overline{Q} = 1$ ), la línea BIT se pone a 0.  
 $\text{BIT} = 0$  y  $\overline{\text{BIT}} = 1 \rightarrow$  se lee un 0

La lectura NO  
es destructiva



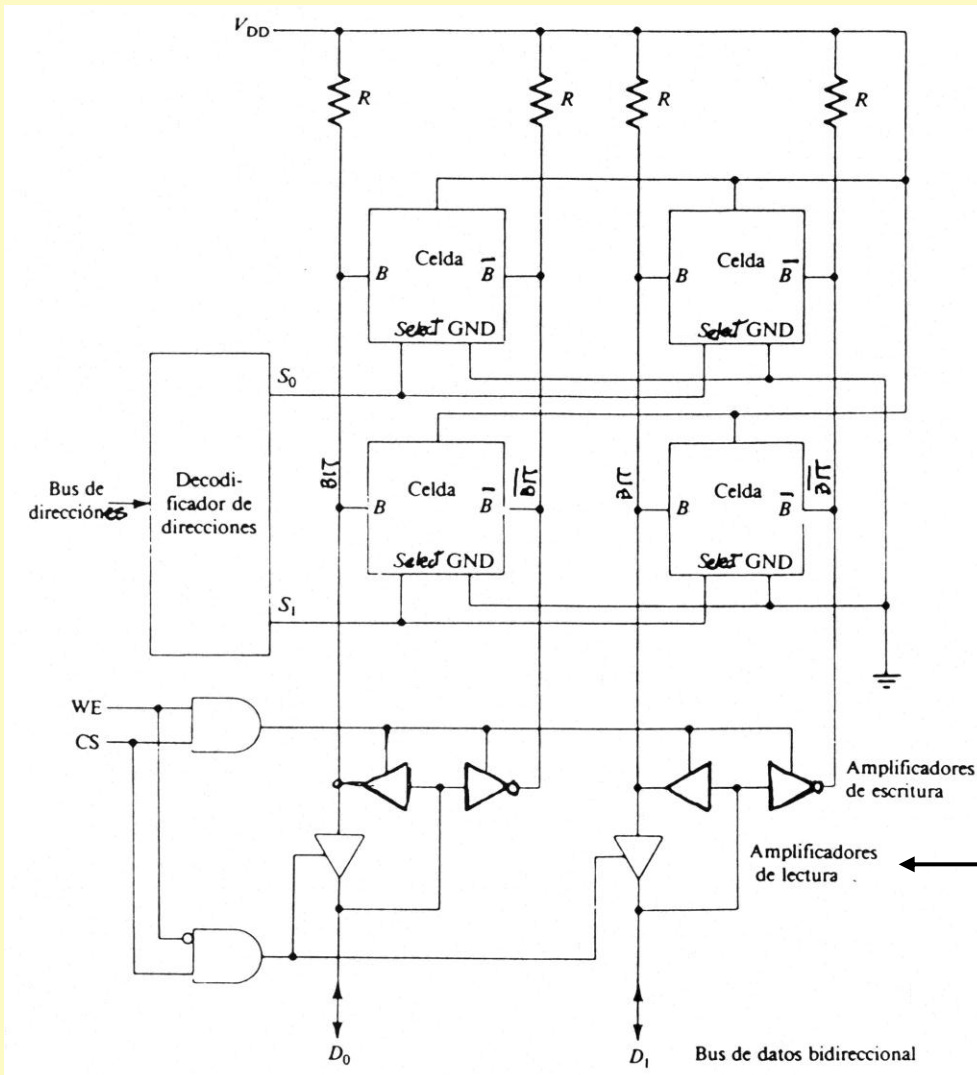
## OPERACIÓN DE ESCRITURA

- Se requieren circuitos de control que fuercen las líneas de bit a valor “0” o “1” ‘*fuertes*’.
- Se selecciona la celda poniendo SELECT o WORD a “1”, igual que en la lectura.
- Si se desea, por ejemplo, escribir un “1” y actualmente  $Q = 0$  ( $\overline{Q} = 1$ ), los circuitos de control aplican un “1” ‘*fuerte*’ en BIT y un “0” ‘*fuerte*’ en  $\overline{BIT}$ . Esta combinación de entradas hace que el biestable de almacenamiento cambie de estado ( $Q = 1, \overline{Q} = 0$ ). Este estado permanecerá en la celda cuando WORD = “0”. Si la celda ya almacenaba un “1” el escribir un “1” no afecta a la celda.
- Para escribir un “0”, BIT y  $\overline{BIT}$  se fuerzan a “0” y “1”, respectivamente.



## ESQUEMA SIMPLIFICADO DE CONEXIÓN DE CELDAS DE UNA SRAM

SRAM de 2 palabras de 2 bits:



ESCRITURA  $\approx$  WE = 1  
LECTURA  $\approx$  WE = 0

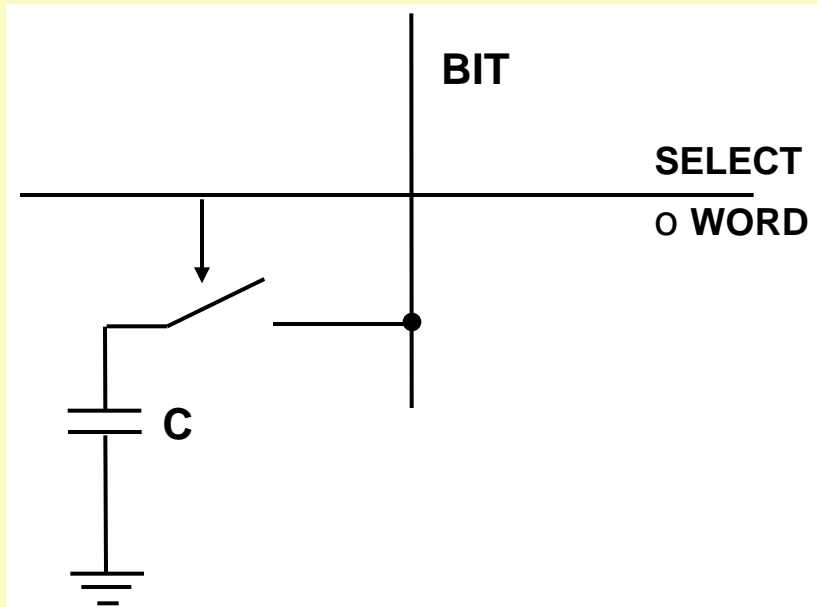
Si CS = 1

(Deshabilitados en Lectura)

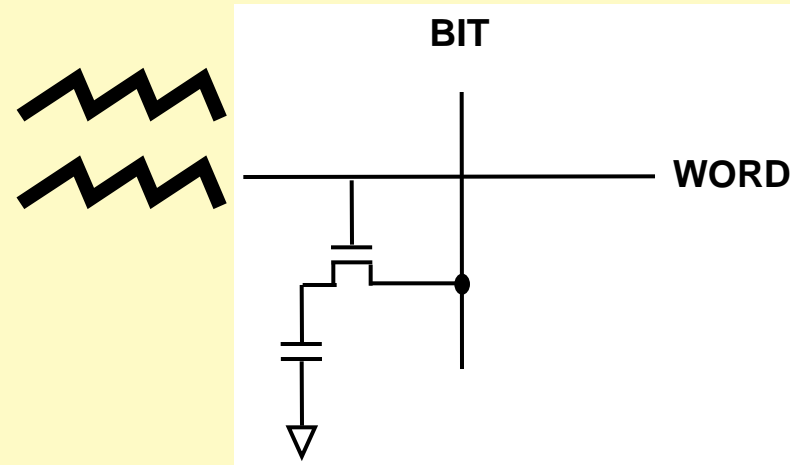
(Deshabilitados en Escritura)



## CELDA DE MEMORIA SRAM



Dinámica → Los datos almacenados decaen o se desvanecen y deben ser restaurados a intervalos regulares.



- La información se almacena en forma de carga eléctrica en una capacidad **C**. La presencia (ausencia) de carga indica que hay almacenado un “1” (“0”).

VENTAJA: Sencillez de la celda de almacenamiento.

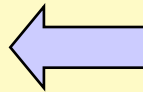
- El transistor actúa como un conmutador para cargar y descargar la capacidad **C** a través de una única línea de bit **BIT**.
- La línea **SELECT** o **WORD** hace conmutar el transistor de corte a conducción o viceversa.



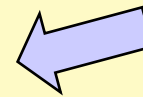
## OPERACIÓN DE LECTURA

- Una circuitería externa convierte a BIT en una línea de salida, seleccionándose la celda con WORD= “1”.
- Si C está cargada (=“1”), entonces se descarga a través de la línea BIT, produciendo un pulso de corriente que es detectado por un amplificador de salida (“SENSE AMPLIFIER”), haciendo que aparezca un “1” en la línea de datos de salida.
- Si C está descargada (=“0”) no se produce pulso de corriente, generándose un “0” en la línea de datos de salida.

Esta etapa extra de escritura la realizan automáticamente los circuitos de control de la DRAM



La operación de lectura debe ir seguida de una operación de escritura que restaure el estado original



La lectura es destructiva



## OPERACIÓN DE ESCRITURA

- Se selecciona BIT como entrada y se pone a “0” o a “1”, seleccionándose la celda con WORD = “1”.
- C se carga si BIT = “1” o se descarga si BIT = “0” .

## OPERACIÓN DE REFRESCO

- El aislamiento de la capacidad C no es perfecto, por lo que la carga almacenada tiende a desvanecerse en unos pocos milisegundos.



Los contenidos de la DRAM deben ser restaurados a intervalos regulares. Puede haber un circuito de control externo que genere secuencialmente todas las direcciones, especificando una operación de lectura para cada dirección.



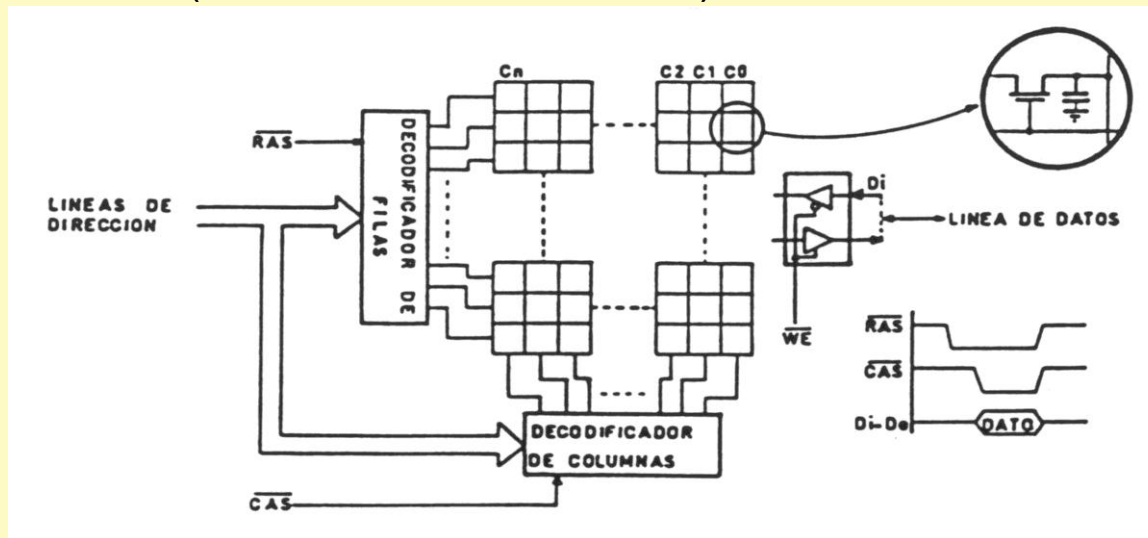


## CIRCUITOS DE MEMORIA DRAM

- La capacidad elevada de los chips de memoria **RAM DINÁMICA** hace que las direcciones tengan que proporcionarse multiplexadas en el tiempo.

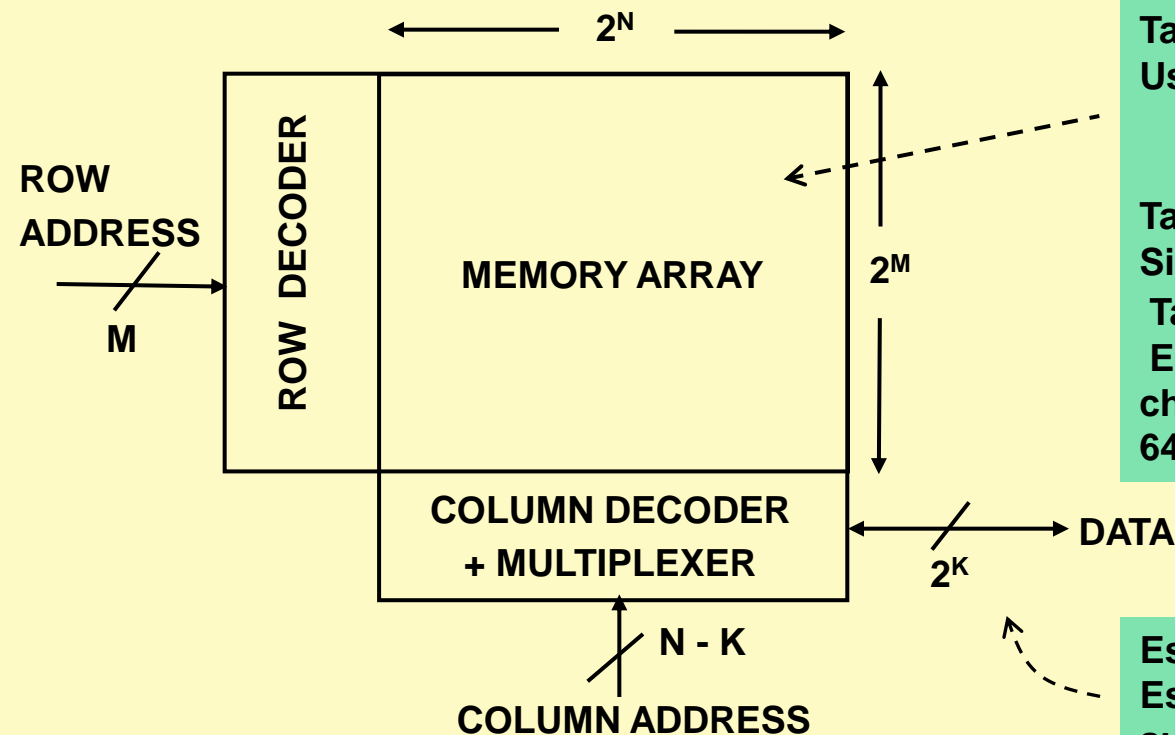
Las celdas de memoria se organizan en una matriz lo más cuadrada posible de cara a minimizar el área de circuito integrado (esto también ocurre en las SRAM).

Los bits de dirección correspondientes a las filas de la matriz de memoria se proporcionan en primer lugar, validándose por la señal  $\overline{\text{RAS}}$  (Row Access Strobe). Luego se proporcionan los bits que direccionan las columnas, validados por  $\overline{\text{CAS}}$  (Column Access Strobe.)





- Para generar las señales  $\overline{RAS}$  y  $\overline{CAS}$ , realizar la interfase entre las señales que genera la CPU y el bus del sistema, y controlar el refresco, se suele utilizar un **CONTROLADOR DE MEMORIA RAM DINÁMICA**.
- No existe la señal CS como en las SRAM sino que para dar por válido un acceso es necesaria la secuencia completa  $\overline{RAS}$ ,  $\overline{CAS}$  junto con la señal  $\overline{WE}$ .



Tamaño =  $2^M * 2^N$   
Usualmente  $M = N = N^{\circ}$  de patillas de dirección del chip.

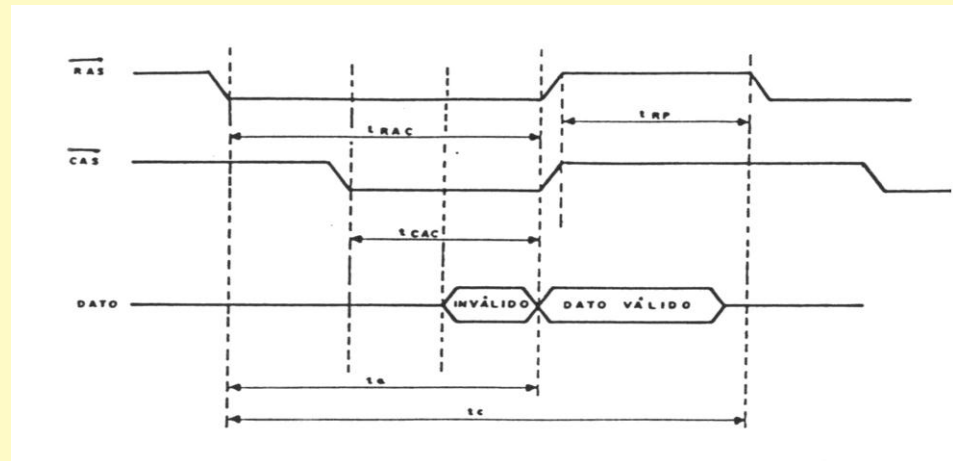
Tamaño =  $(2^M)^2 = 2^{2M}$   
Si aumentamos el n° de patillas en 1 →  
Tamaño =  $(2^{M+1})^2 = 2^{2M+2} = 4 * 2^{2M}$  →  
El tamaño crece 4 veces, por eso los chips crecen:  
64K → 256K → 1M → 4M → 16M → 64M ...

Es usual que  $k = 0 \rightarrow 2^k = 1$   
Es decir, los chips DRAM suelen almacenar un solo bit, en lugar de bytes o palabras.



- Acceso a una memoria DRAM:

(LECTURA)



$t_a$   $\approx$  **TIEMPO DE ACCESO**: Tiempo desde que se inicia una lectura hasta que el dato está disponible en el bus de datos.

Coincide con  $t_{RAC}$   $\approx$  Tiempo de acceso desde  $\overline{RAS} = "0"$

No se puede comenzar un segundo acceso tras  $t_a$ , dado que las lecturas son destructivas (el condensador asociado a la celda de memoria se descarga) y es necesario esperar un **Tiempo De Recarga De Fila** ( $t_{RP}$ ) para que la circuitería interna de la DRAM restaure el valor de las celdas de la fila correspondiente.

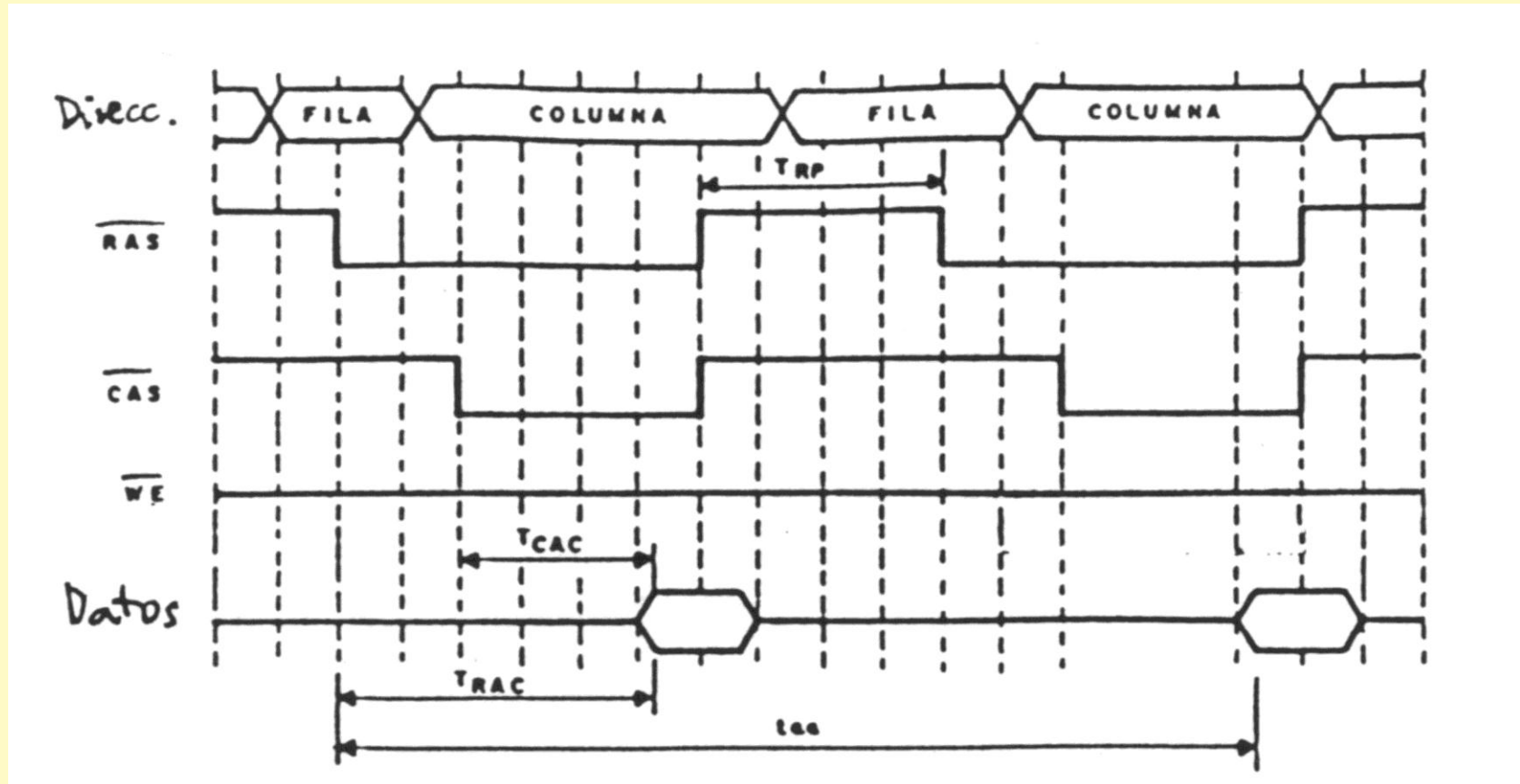
$t_c$   $\approx$  **TIEMPO DE CICLO**: Tiempo mínimo entre dos accesos consecutivos.

$$t_c = t_{RAC} + t_{RP} \quad ; \quad t_c < t_a$$

(En una memoria estática coinciden  $t_a$  y  $t_c$ )



- Acceso a dos palabras en un chip DRAM:



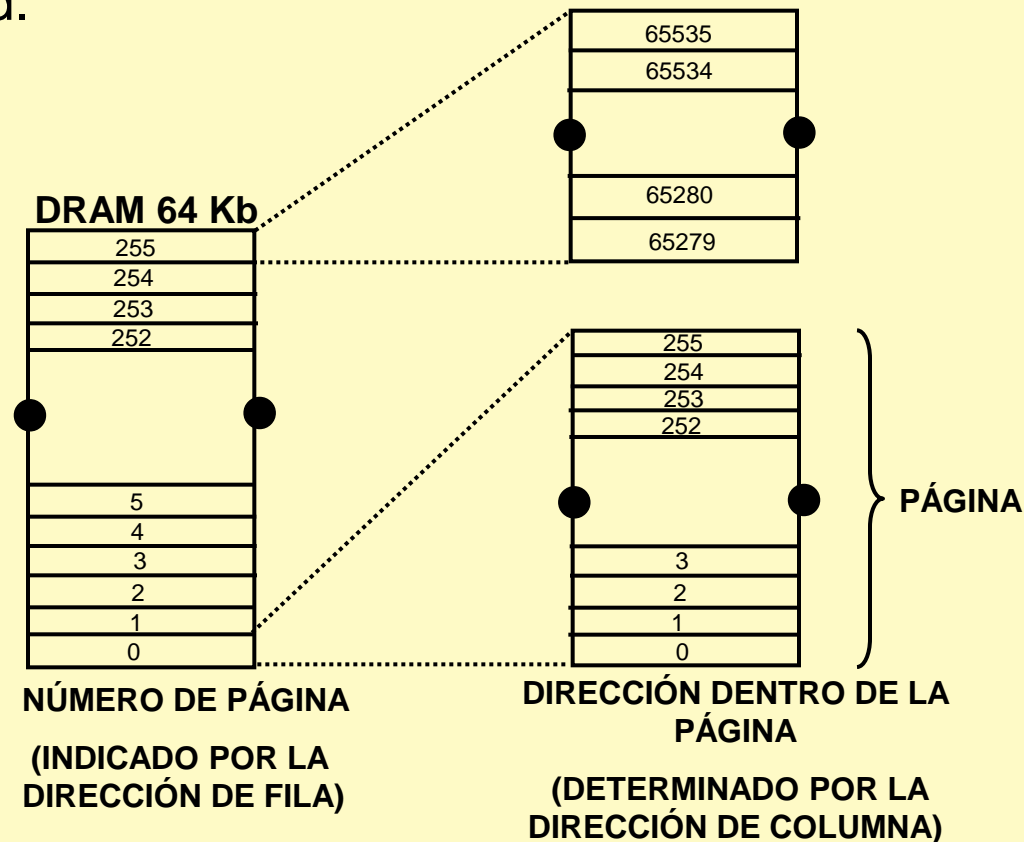
$$t_{aa} = t_{RAC} + t_{RP} + t_{RAC}$$

$$(Ej: t_{aa} = 80ns + 60ns + 80ns = 220 ns)$$




- Acceso en modo de página:

Se puede acceder a cualquier dirección (columna) dentro de una página (fila) una vez seleccionada ésta última manteniendo  $\overline{RAS}$  a "0" y cambiando  $\overline{CAS}$  de "1" a "0" en cada acceso. Con esto se consigue una mayor velocidad.





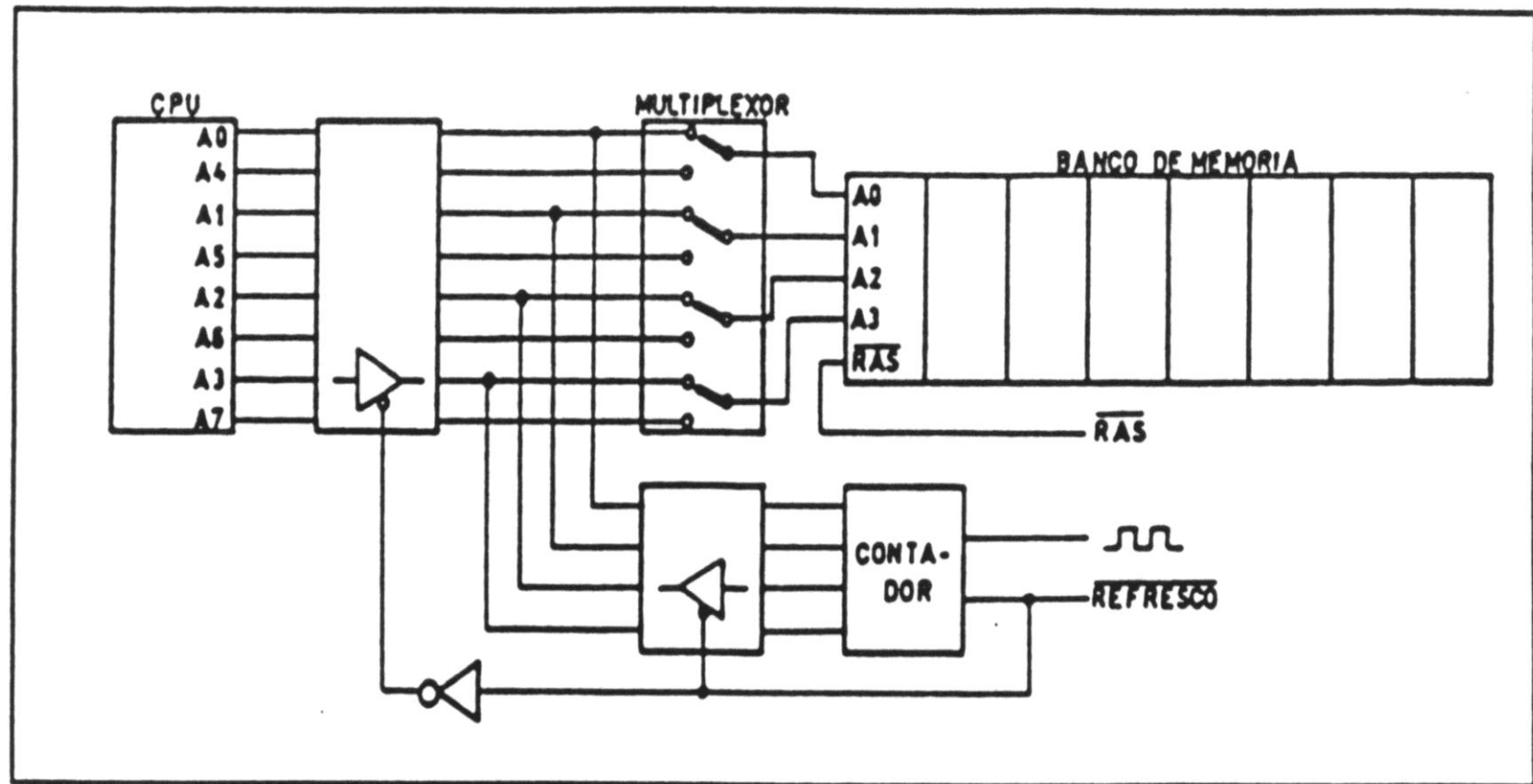
- Para realizar el refresco de memoria se precisa una circuitería auxiliar que produzca **ciclos de refresco**, consistentes en recargar los condensadores asociados a todas las celdas.  Deben producirse cada varios ms (de 2 a 8ms, aproximadamente)

El ciclo de refresco consiste en **dar un impulso  $\overline{\text{RAS}}$  junto con una dirección de fila para todas las filas**. Con eso se refrescan todas las columnas de cada fila.

La CPU no puede hacer uso de la memoria durante un ciclo de refresco ni emplear, por tanto, los buses. Estos serán controlados por la circuitería auxiliar basada fundamentalmente en un **contador**.



- ★ Controlador de DMA
- ★ Circuito Controlador de memoria DRAM



*Esquema genérico del circuito de Refresco*



- Cuando se recibe una orden de refresco (dada periódicamente por un temporizador) la CPU se desconecta de los buses.  
El contador genera todas las posibles direcciones de fila y al mismo tiempo va aplicando la señal  $\overline{RAS}$ .  
Finalizado el ciclo de refresco el contador se desconecta de los buses, tomándolos de nuevo la CPU.
- Otra forma de hacer el refresco consiste en refrescar **una sola fila cada vez**. Así la CPU está menos tiempo sin los buses, pero los refrescos se deben pedir con más periodicidad.
- Los modernos chips DRAM contienen **sus propios contadores de refresco** y modos de refresco que no necesitan intervención de circuitería externa.
- Aparte de los ciclos de refresco, los chips DRAM refrescan automáticamente la fila accedida en cualquier ciclo de lectura o escritura.





1.Introducción

3.Jerarquía de Memoria

3. Memorias de Semiconductores

4. Configuración y Diseño de Memorias

5. Memorias Asociativas

6. Memorias Caché

## TEMA 6: ORGANIZACIÓN DE MEMORIA

1. Introducción

2. Jerarquía de Memoria

3. Memorias de Semiconductores

➔ **4. Configuración y Diseño de Memorias**

5. Memorias Asociativas

6. Memorias Caché



## ➔ Configuración y Diseño de Memorias

1. Incrementar el Ancho de Palabra.
2. Incrementar el Número de Palabras.
3. Incrementar Simultáneamente Ancho y Número.
4. Ejemplos de Diseño.

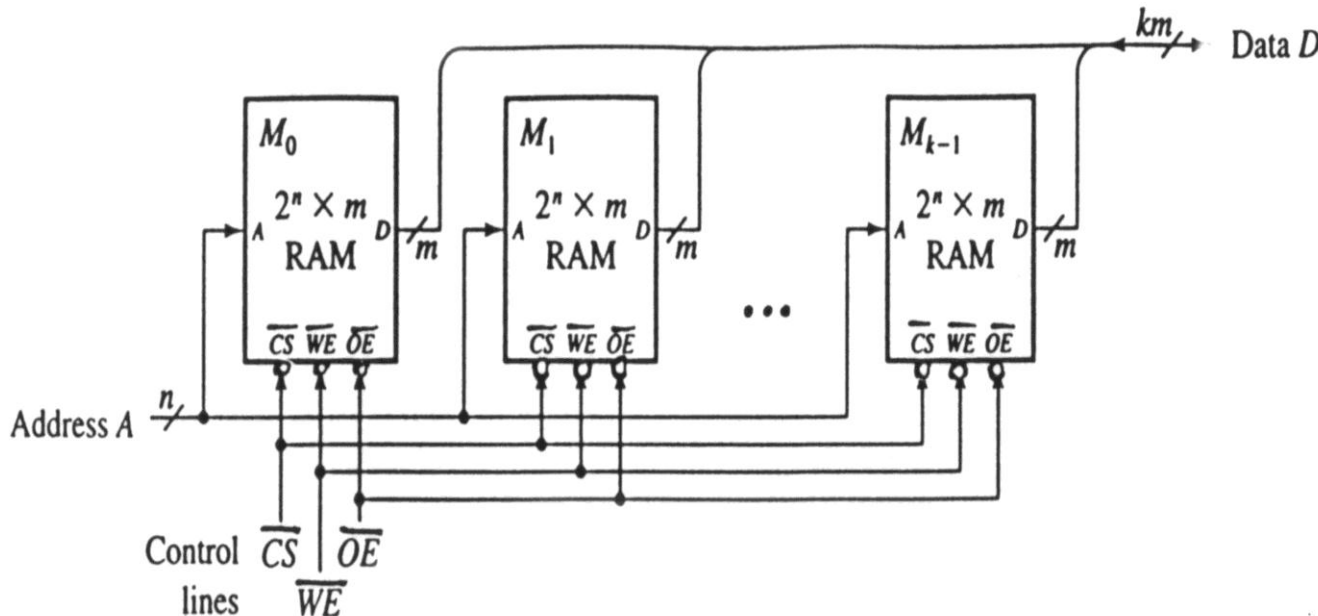


## CONFIGURACIÓN Y DISEÑO DE MEMORIAS UTILIZANDO VARIOS CHIPS

**PROBLEMA:** Construir una memoria de  $2^N$  palabras de  $M$  bits a partir de chips de  $2^n$  palabras de  $m$  bits.

INCREMENTAR EL ANCHO DE PALABRA DE  $m$  A  $k \cdot m = M$

Necesitaremos  $k$  circuitos de tamaño de palabra  $m$ :



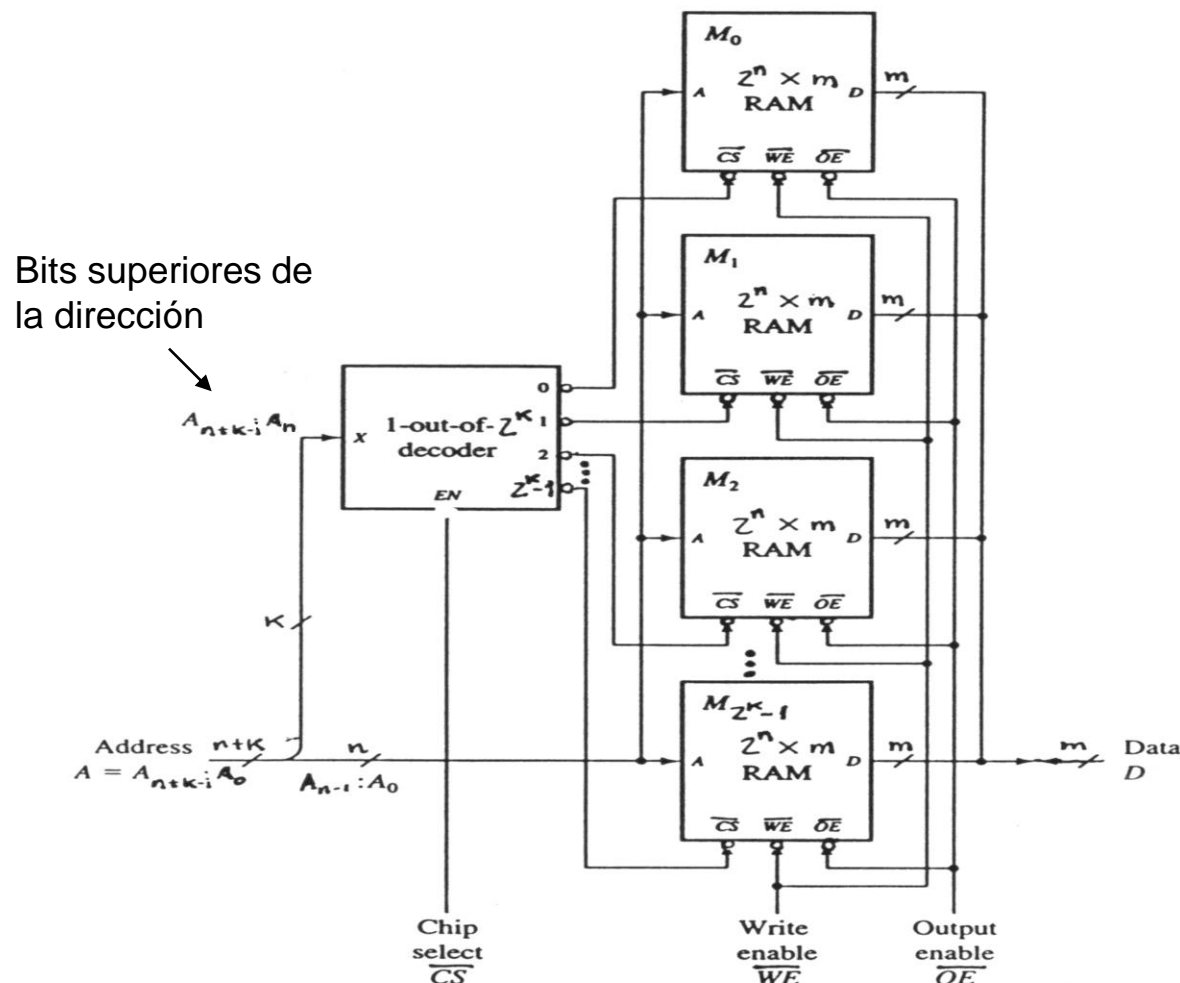
Los  $k$  chips se conectan de una forma “bit - slice” con cada chip contribuyendo a  $m$  bits de la palabra de datos de  $k \cdot m$  bits.

Las líneas de control se conectan a todos los chips



## INCREMENTAR EL NÚMERO DE PALABRAS DE $2^n$ A $2^{n+k} = 2^N$

Necesitaremos  $2^k$  circuitos de  $2^n$  palabras y un decodificador de 1 entre  $2^k$ :



Las  $n$  líneas de dirección de orden inferior se conectan en común a los terminales de entrada de direcciones de los  $2^k$  chips.

Las  $k$  líneas de dirección nuevas de orden más alto  $A_{n+k-1}:A_n$  se conectan a las entradas de un circuito decodificador de  $k$  a  $2^k$ .

Las  $2^k$  líneas de salida de este decodificador se conectan a las entradas de selección  $\overline{CS}$  de los  $2^k$  circuitos.

Cada configuración de los  $n+k$  bits hace que se seleccione solamente el circuito RAM indicado por los bits de dirección  $A_{n+k-1}:A_n$ .

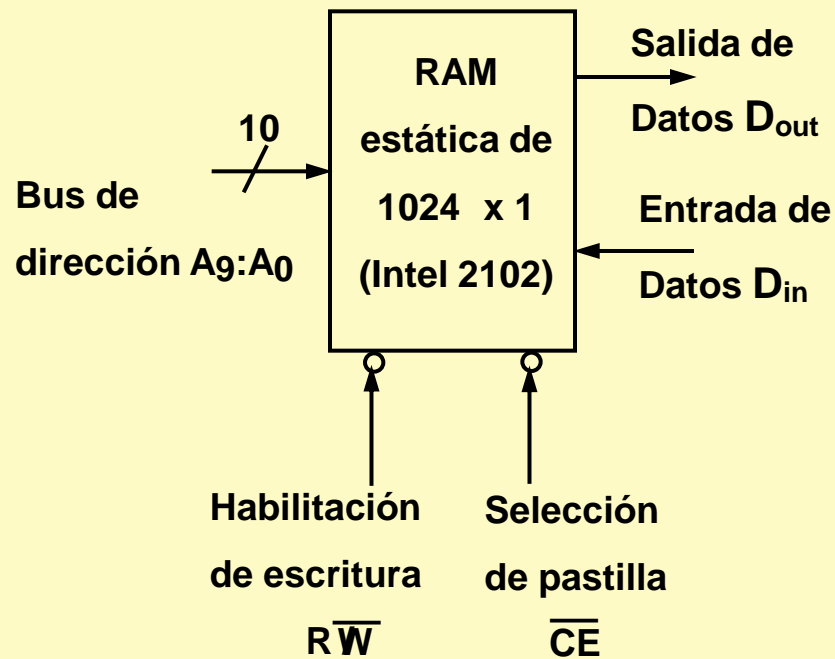
Las líneas de datos se unen gracias a que cuando  $\overline{CS} = 1$  en un circuito RAM, sus líneas de datos pasan al estado de alta impedancia.



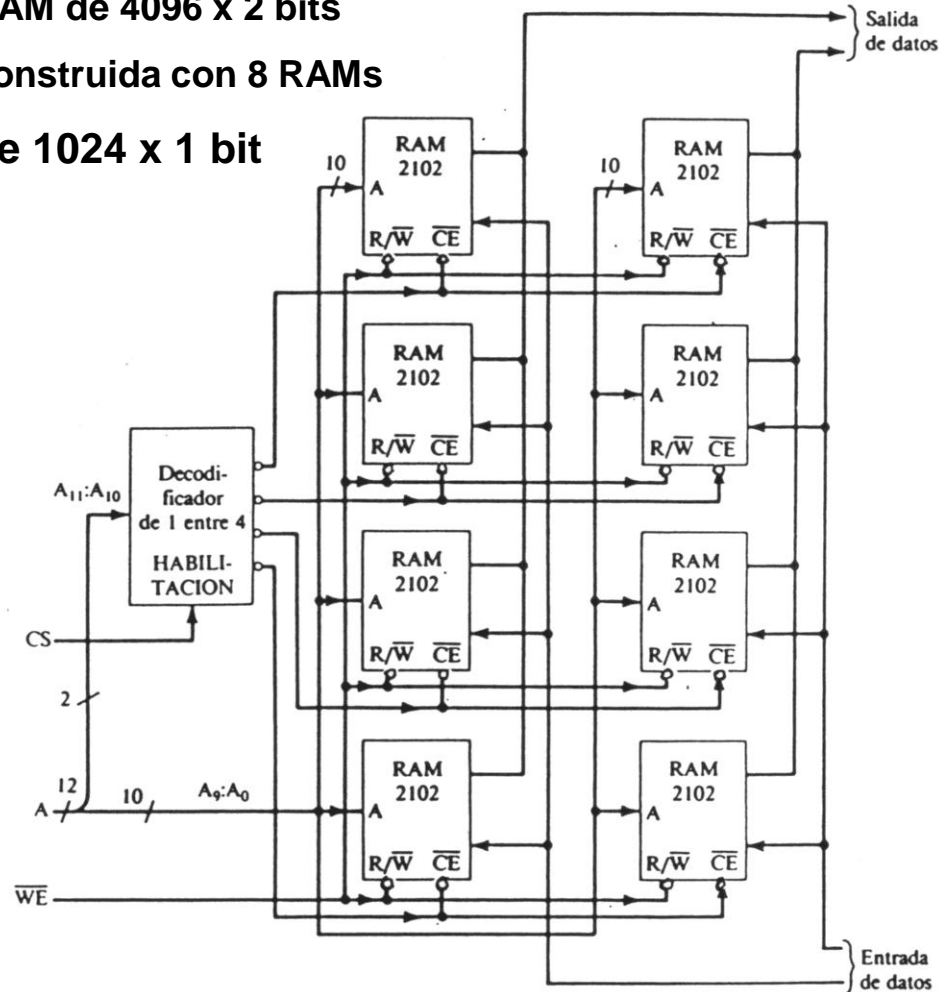
## INCREMENTAR SIMULTÁNEAMENTE EL NÚMERO DE PALABRAS Y EL ANCHO DE BANDA

Basta combinar las dos técnicas anteriores.

### EJEMPLO 1:

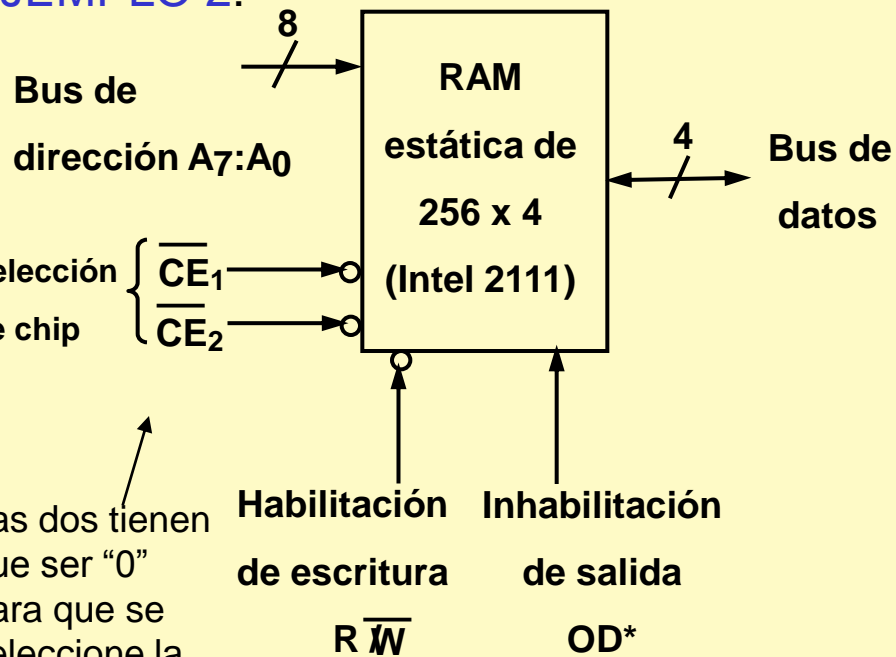


RAM de 4096 x 2 bits  
construida con 8 RAMs  
de 1024 x 1 bit



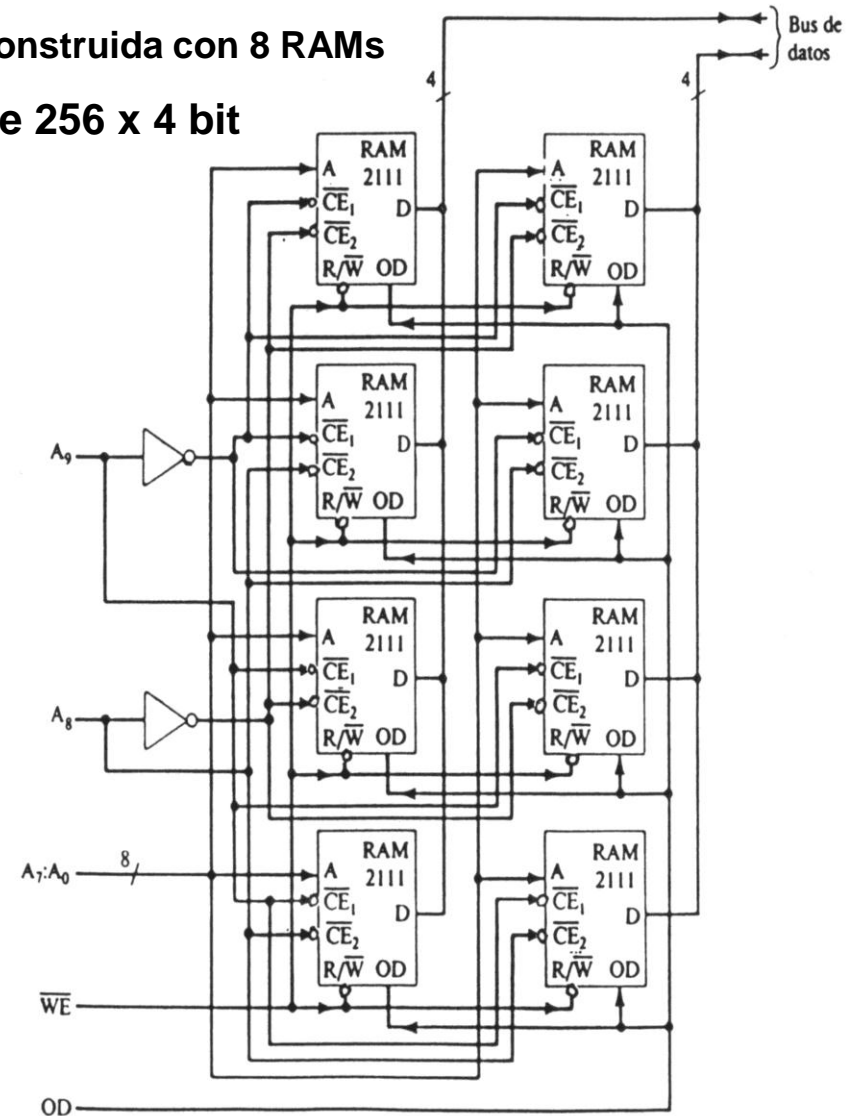


## EJEMPLO 2:



(\*)Output Disable: Como hay un solo bus de datos, la línea OD se activa durante los ciclos de escritura para evitar que se lean datos internos desde el chip al bus de datos mientras éste se está utilizando como bus de entrada.

RAM de 1 Kbyte  
construida con 8 RAMs  
de 256 x 4 bit



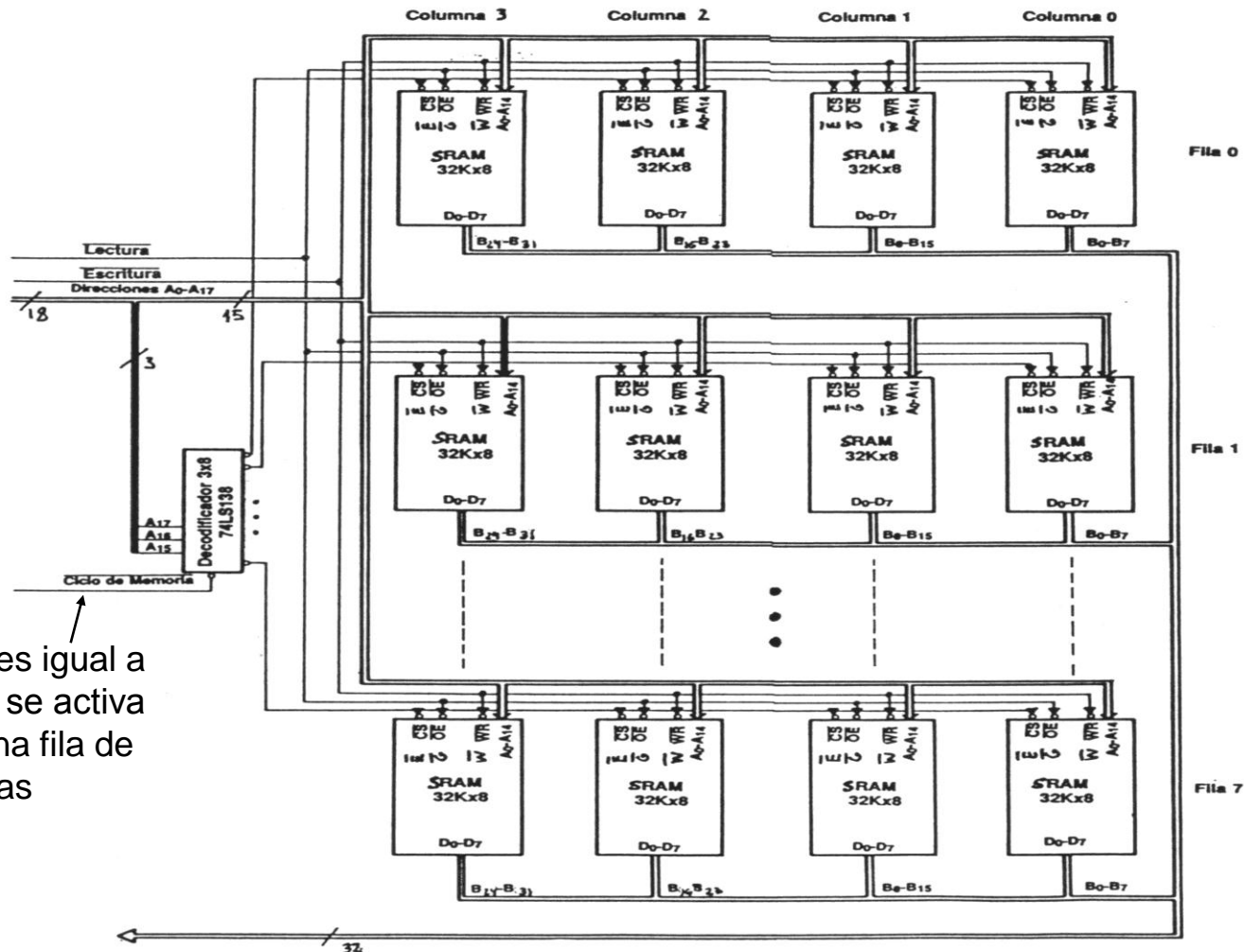


## EJEMPLOS DE DISEÑO

### EJEMPLO 1: SRAM de 256 K Palabras de 32 bits

Se requieren 8 filas de 4 pastillas = 32

Empleando pastillas de 32K x 8 bits pastillas



Si no es igual a "1" no se activa ninguna fila de pastillas

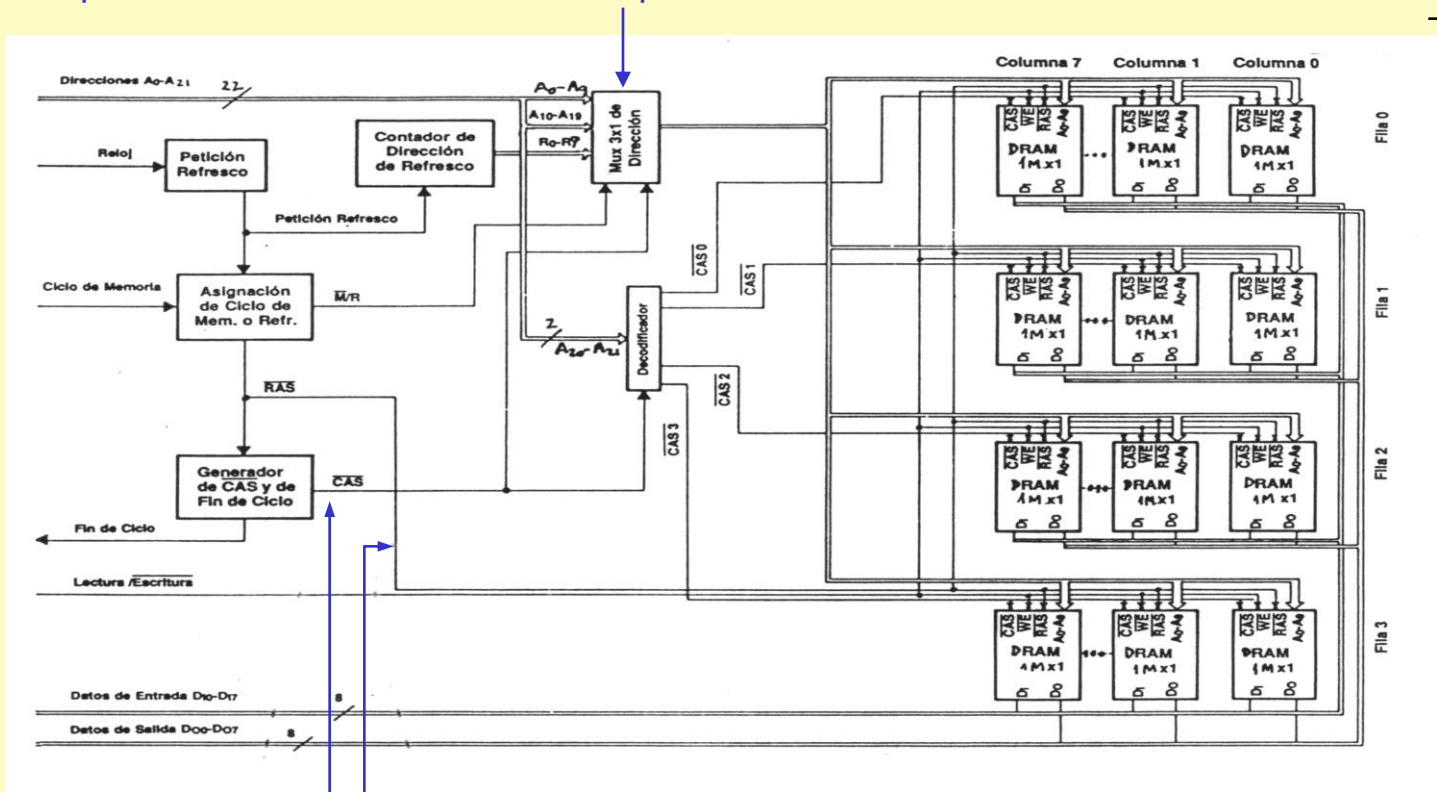




## EJEMPLO 2: DRAM de 4Mbytes

Empleando pastillas de 1M x 1 bit.

El multiplexor permite enviar a las pastillas la parte inferior o superior de los 16 bits de dirección que necesitan



$\overline{M/R}$	$\overline{CAS}$	Direcciones
0	0	$A_0 - A_9$
1	0	$A_{10} - A_{19}$
1	X	$R_0 - R_9$

La señal  $\overline{RAS}$ , que especifica la primera mitad de la dirección, es común a todas las pastillas.

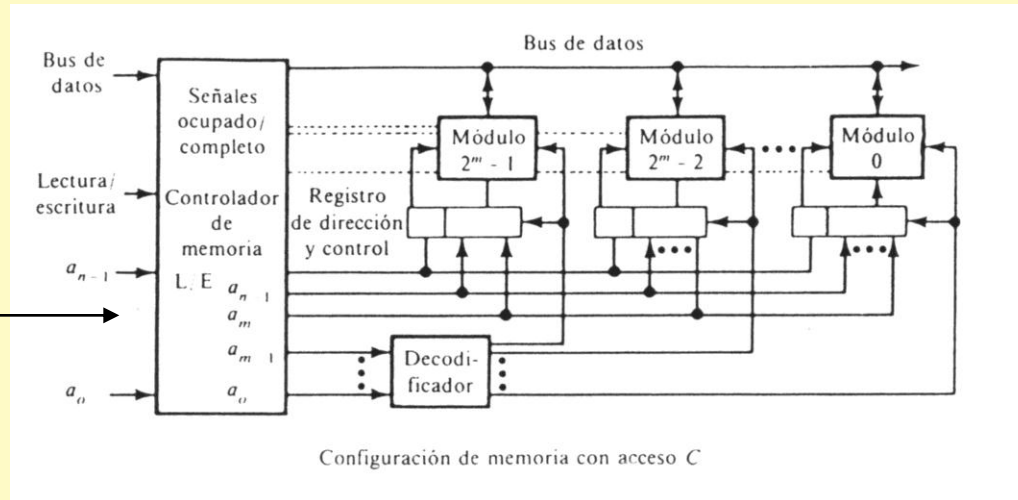
La señal  $\overline{CAS}$  además de para especificar la otra mitad, sirve como  $\overline{CS}$ , se activa de forma independiente para cada fila de pastillas.



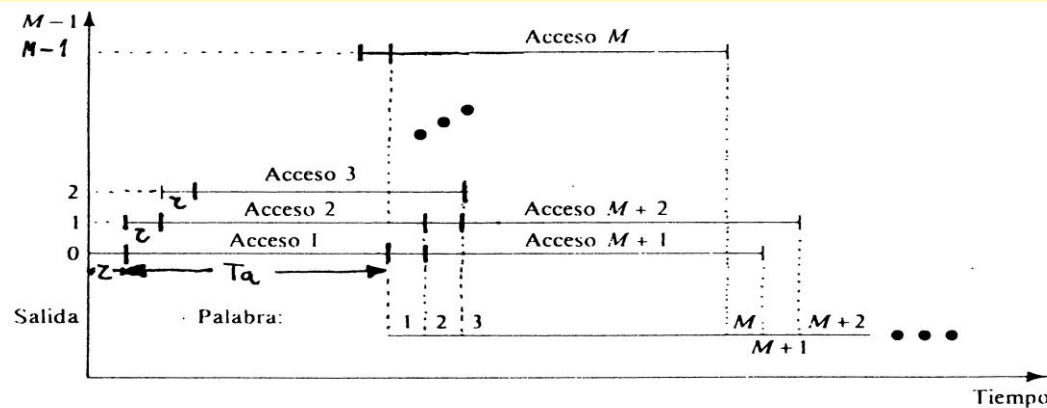


- Acceso concurrente tipo “C” o con latch en la entrada:

El controlador de memoria se usa para retener una petición que reference un módulo ocupado e iniciar el acceso cuando el módulo complete su ciclo actual.



En tiempo  $\tau$  se escribe la dirección dentro de un módulo en su latch de entrada. Tras  $T_a$  la palabra está disponible en el bus de datos.



Latch en la entrada → CADA MÓDULO PUEDE USAR UNA DIRECCIÓN PARTICULAR



- 1.Introducción
- 3.Jerarquía de Memoria
- 3. Memorias de Semiconductores
- 4. Configuración y Diseño de Memorias
- 5. Memorias Asociativas
- 6. Memorias Caché

## TEMA 6: ORGANIZACIÓN DE MEMORIA

1. Introducción
2. Jerarquía de Memoria
3. Memorias de Semiconductores
4. Configuración y Diseño de Memorias
- ➔ **5. Memorias Asociativas**
6. Memorias Caché



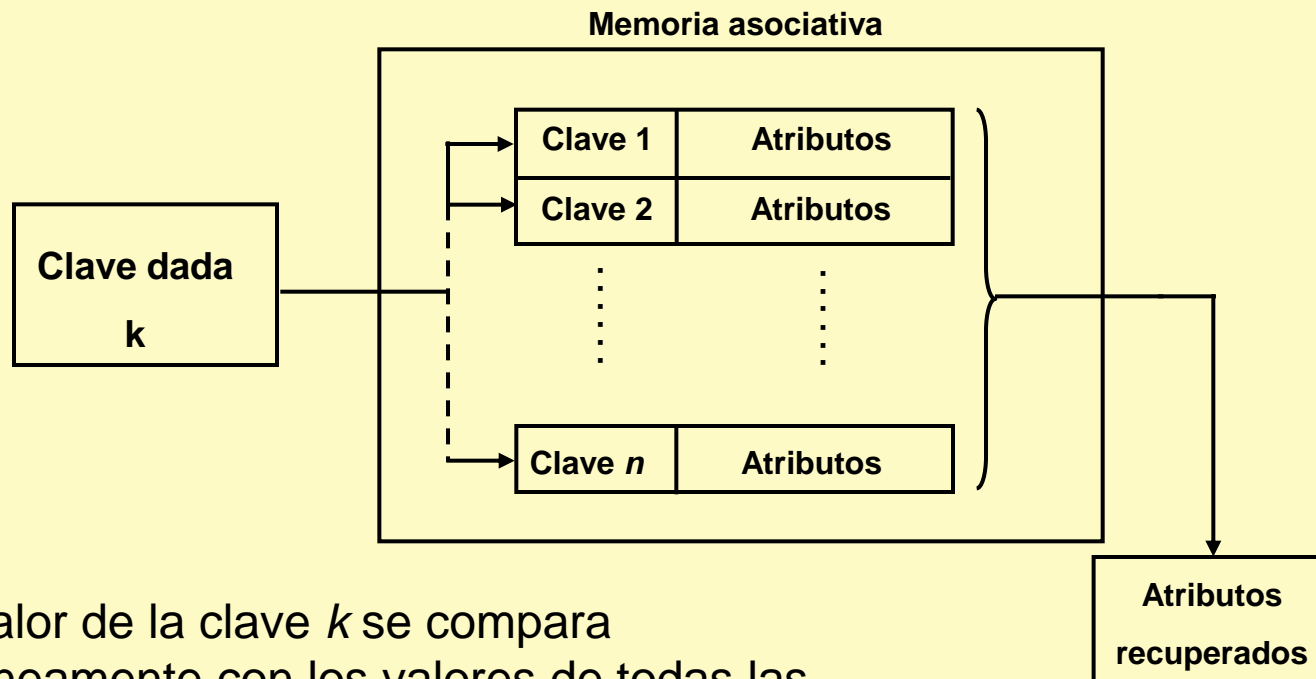
## → Memorias Asociativas

1. Operaciones.
2. Estructura.
3. Lógica de Enmascaramiento y de Comparación.
4. Lectura.
5. Escritura.

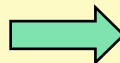


## MEMORIAS ASOCIATIVAS

- También llamadas:
- MEMORIAS DIRECCIONABLES POR SU CONTENIDO ( CAM  $\approx$  Content Addressable Memory )
  - MEMORIAS DE BÚSQUEDA PARALELA
  - MEMORIAS MULTIACCESO
- Estructura hardware que permite efectuar **operaciones de acceso a los datos (búsqueda, comparación) a gran velocidad.**
  - Representación conceptual de la memoria asociativa:



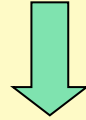
- El valor de la clave  $k$  se compara simultáneamente con los valores de todas las claves almacenadas en la memoria



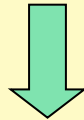
**Si alguna de ellas coincide, se leen los datos correspondientes (Atributos).**



- El **tiempo de búsqueda** es **muy pequeño**, ya que el hardware efectúa todas las comparaciones a la vez.
- ! La búsqueda se efectúa sólo a partir del contenido de la clave, y NO de la dirección de la misma.



La información ha de estar en una matriz de memoria diseñada específicamente para permitir ser accedida por contenido.



Su principal inconveniente es un **coste hardware mucho mayor** que el de una RAM.



## EJEMPLO :

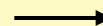
Nombre	Peso	Talla	Edad
P. Pérez	63	1.71	27
M. García	90	1.87	45
M. Ortega	82	1.83	33
A. Álvarez	51	1.64	61

← Tabla  
almacenada en memoria

- Si utilizamos una memoria de acceso aleatorio convencional necesitamos especificar una dirección física para poder acceder a su información asociada.



**Esta dirección física no tiene  
ninguna relación lógica con la  
información que almacena.**



**EL MÉTODO DE  
ACCESO ES ARTIFICIAL Y  
COMPLICA LA PROGRAMACIÓN**

Por ejemplo, para ver quién mide 1.83, hay que realizar una búsqueda



**Lentitud de la búsqueda convencional**



- ALTERNATIVA: Emplear uno de los campos de la tabla como clave para acceso al registro/s que contiene/n esa clave. (MEMORIA ASOCIATIVA)

***EN GENERAL, UNA MEMORIA ASOCIATIVA TIENE LA CAPACIDAD DE ACCEDER A UNA PALABRA ALMACENADA USANDO COMO DIRECCIÓN SU CONTENIDO O EL CONTENIDO DE UN SUBCAMPO DE LA MISMA.***



## OPERACIONES CON UNA MEMORIA ASOCIATIVA (Pensadas para Bases de Datos)

- BÚSQUEDAS EXTREMALES
  - *Valor máximo*
  - *Valor mínimo*
  - *Valor medio*
- BÚSQUEDAS DE EQUIVALENCIA
  - *Igual a*
  - *No igual a*
  - *Similar a* (Valores idénticos en varios campos a uno dado)
  - *Próximo a* (Valores que satisfacen una cierta condición de proximidad)
- BÚSQUEDAS POR UMBRAL
  - *Menor que*
  - *Mayor que*
  - *No menor que*
  - *No mayor que*
- BÚSQUEDAS ENTRE LÍMITES *acotadas a un cierto intervalo (abierto/cerrado)*
- EXTRACCIONES ORDENADAS
  - *Ascendentemente*
  - *Descendentemente*

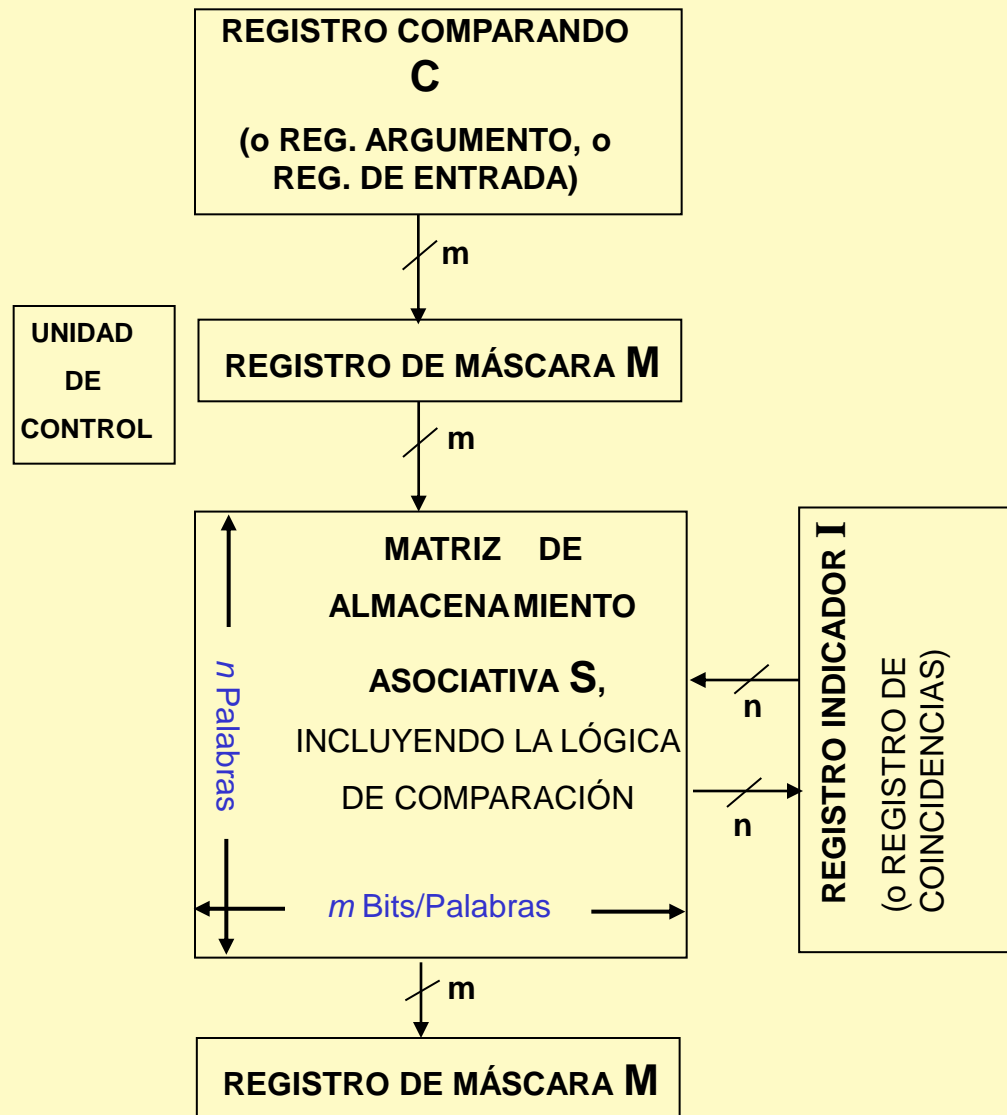
### APLICACIONES:

- Implementación de memorias caché asociativas,
- Implementación de TLBs (Buffers de Traducción Anticipada) en memoria virtual,
- Manipulación de información almacenada en Bases de Datos, etc.





## ESTRUCTURA DE LA MEMORIA ASOCIATIVA



### REGISTRO DE ENTRADA O COMPARANDO

Operando que sirve de clave de búsqueda o comparación, o que contiene la información que se va a escribir.

### REGISTRO DE MÁSCARA

Selecciona los bits del registro de entrada que intervendrán en la búsqueda en paralelo. Un bit del registro de entrada se utilizará en el proceso de búsqueda sólo si el correspondiente bit del registro de máscara está a 1.

### REGISTRO DE SALIDA

Contiene una de las palabras (o la suma lógica de varias) seleccionada por el proceso de búsqueda.

### REGISTRO DE COINCIDENCIAS O INDICADOR

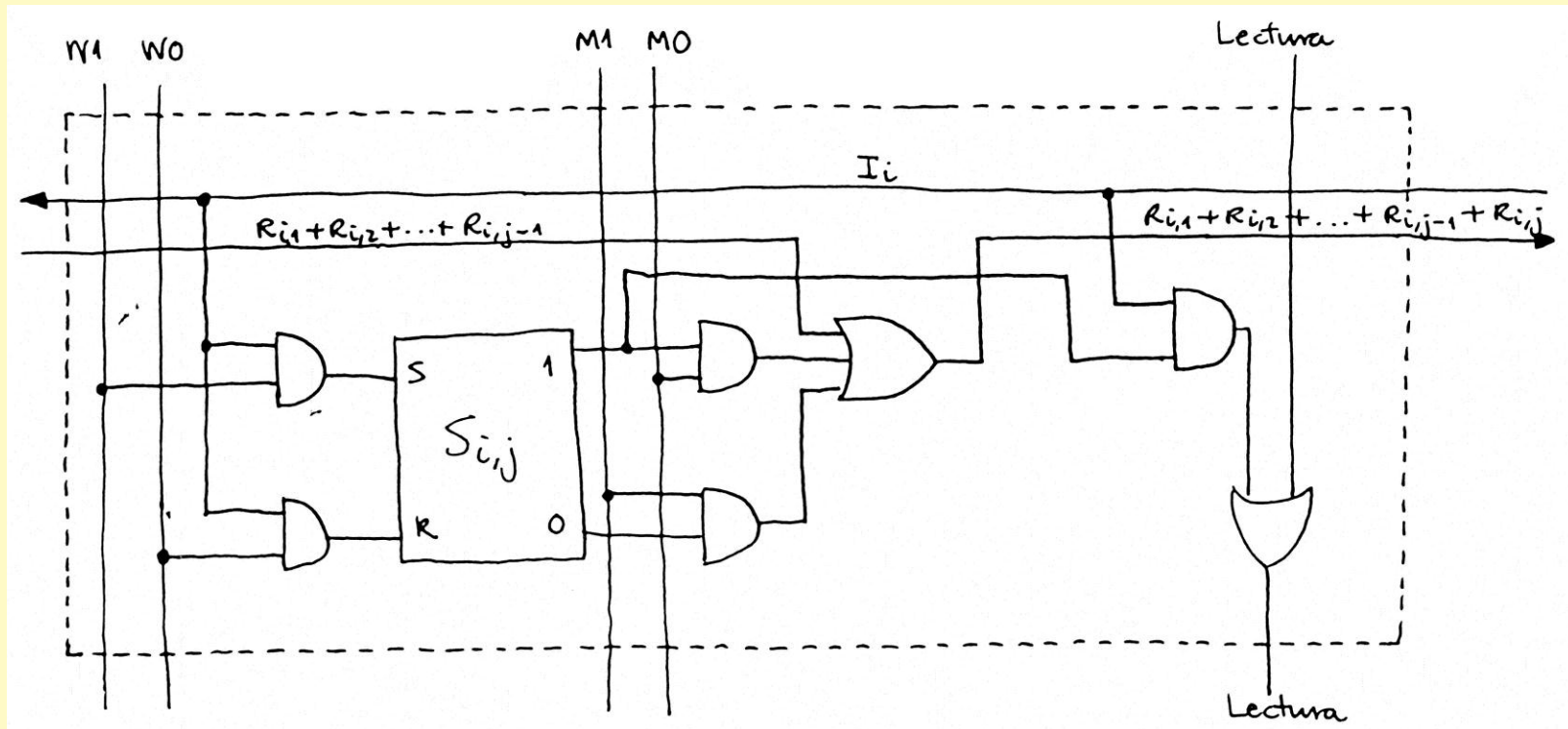
Recibe el resultado generado por la lógica de comparación. Cada bit contiene el resultado de la comparación con una palabra. Puede estar acompañado de uno o más registros temporales para almacenamiento de resultados de comparaciones previas.

**MATRIZ ASOCIATIVA:** Colección de celdas básicas de memoria  $S_{ij}$  ( $1 \leq i \leq n$ ,  $1 \leq j \leq m$ ); cada una con un bit.

**UNIDAD DE CONTROL:** Da las órdenes de comparación, lectura y escritura.



## ESTRUCTURA GLOBAL DE UNA CELDA DE UN BIT





- 1.Introducción
- 3.Jerarquía de Memoria
- 3. Memorias de Semiconductores
- 4. Configuración y Diseño de Memorias
- 5. Memorias Asociativas
- 6. Memorias Caché

## TEMA 6: ORGANIZACIÓN DE MEMORIA

- 1. Introducción
- 2. Jerarquía de Memoria
- 3. Memorias de Semiconductores
- 4. Configuración y Diseño de Memorias
- 5. Memorias Asociativas
- ➔ **6. Memorias Caché**



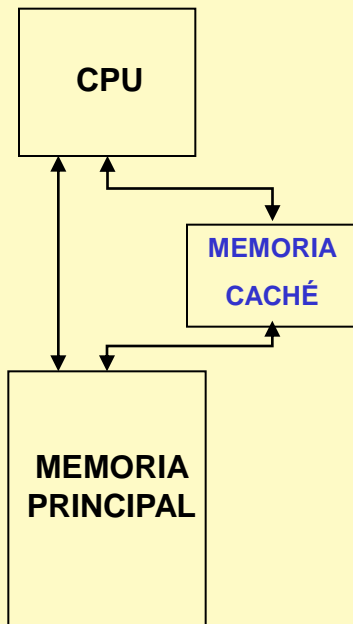
## → Memorias Cachés

1. Objetivos.
2. Modelo para Evaluar las Prestaciones.
3. Estructura de un Sistema Caché Típico.
4. Operación de una Caché.
5. Aspectos del Diseño.
6. Políticas de Extracción.
7. Políticas de Colocación.
8. Políticas de Reemplazo.
9. Políticas de Actualización.
10. Cachés Separadas Datos/Instrucciones.
11. Diseño del Sistema de Memoria para soportar Cachés.
12. Ejemplo: Unidad de Caché Interna del 486.



## MEMORIAS CACHÉ

- Memoria pequeña y rápida que se sitúa entre el procesador y la memoria principal.<sup>(1)</sup> 8K – 256K. 5 – 10 veces más rápida que la MP
  - Nivel más elevado de la jerarquía de memoria
  - 1ª Caché: IBM 360/85, 2ª mitad de los 60.
- En la caché se almacenan aquellas palabras de la MP actualmente en uso.
  - El procesador buscará la información que desea en la caché.



- Se encuentra en la caché → **ACIERTO** o **“CACHE HIT”**  
Los datos se transfieren de la caché a la CPU.
- No se encuentra en la caché → **FALTA DE BLOQUE** o **“CACHE MISS”**

Se busca en la MP. Una copia va a la CPU y otra a caché (con su bloque) sustituyéndose un bloque antiguo si no hay sitio.

<sup>(1)</sup> También pueden insertarse entre la memoria principal y la memoria masiva.



- Éxito de la memoria caché: Debido a la propiedad de localidad de los programas.



La información que se va a utilizar en un futuro próximo probablemente ya se encuentre en caché

- OBJETIVOS DE DISEÑO DE UN SISTEMA CON MEMORIA CACHÉ:
  1. **MAXIMIZACIÓN DEL ÍNDICE DE ACIERTOS “A”** (Probabilidad de encontrar la información deseada en la caché) O MINIMIZACIÓN DE  $F = 1 - A$
  2. **MINIMIZACIÓN DEL TIEMPO DE ACCESO “ $t_c$ ” A LA INFORMACIÓN ALMACENADA EN LA CACHÉ.**
  3. **MINIMIZACIÓN DEL RETARDO DEBIDO A UNA FALTA.**
  4. **MINIMIZACIÓN DE LA PENALIZACIÓN POR LA ACTUALIZACIÓN DE LA MP**



## MODELO PARA EVALUAR LAS PRESTACIONES DE UN SISTEMA QUE UTILIZA MEMORIA CACHÉ

$t_c \approx$  TIEMPO DE ACCESO A LA MEMORIA CACHÉ

$A \approx$  RAZÓN DE ACIERTO (HIT RATIO) DE LA MEMORIA CACHÉ

$t_m \approx$  TIEMPO DE ACCESO A LA MEMORIA PRINCIPAL

Tiempo medio de acceso:

$$\overline{T} = \underbrace{A t_c}_{\substack{\text{ACIERTO} \rightarrow \text{No se accede} \\ \text{a la MP}}} + \underbrace{(1 - A)(t_c + t_m)}_{\substack{\text{FALLO} \rightarrow \text{Se accede a la} \\ \text{caché y a la MP}}}$$

$\gamma \approx$  RAZÓN ENTRE LOS TIEMPOS DE ACCESO A LA MP Y A LA CACHÉ

$$\gamma = \frac{t_m}{t_c}$$

EFICIENCIA DE UN SISTEMA QUE EMPLEA MEMORIA CACHÉ:

Eficiencia  $E = \frac{t_c}{\overline{T}} \quad 0 < E \leq 1$

$$E = \frac{t_c}{A t_c + (1 - A)(t_c + t_m)} = \frac{1}{A + (1 - A)(1 + t_m/t_c)} = \frac{1}{A + (1 - A)(1 + \gamma)}$$

$$= \frac{1}{A + 1 - A + \gamma(1 - A)} = \frac{1}{1 + \gamma(1 - A)}$$

E es máxima cuando  $A=1$   
(todas las referencias se encuentran en caché)

$$\gamma \uparrow \rightarrow E \downarrow$$

$$A \downarrow \rightarrow E \downarrow$$

→ Es importante  $\gamma$  baja y  $A$  alta

EJEMPLO:  $t_c = 15 \text{ ns}$   $t_m = 100 \text{ ns}$   $A = 0.9$

$$\left. \begin{aligned} \overline{T} &= A t_c + (1 - A)(t_c + t_m) = \\ &= 0.9 * 15 + 0.1 * 115 = 25 \text{ ns} \\ \gamma &= t_m / t_c = 100/15 = 6.\widehat{6} \end{aligned} \right\} E = \frac{1}{1 + \gamma(1 - A)} = \frac{1}{1 + 6.\widehat{6} * 0.1} = 0.6$$





## ESTRUCTURA DE UN SISTEMA CACHÉ TÍPICO

Consta de dos partes:

- **ZONA DE ALMACENAMIENTO.** MEMORIA RAM ESTÁTICA.
  - Particionada en un conjunto de BLOQUES o LÍNEAS.
  - **BLOQUE o LÍNEA:** Unidad básica de información que se puede transferir entre la memoria caché y la memoria principal.
  - Cada bloque de MP va a un **MARCO DE BLOQUE** en memoria caché.
  - La MP tiene un gran número de bloques y la caché tiene un pequeño número de marcos de bloque → La zona de almacenamiento de la caché mantiene copia de un cierto número de bloques, sin ningún orden particular normalmente.
- **DIRECTORIO CACHÉ.**
  - Usualmente se implementa mediante una o varias memorias asociativas.
  - Guarda las direcciones de los bloques (“**MARCAS**”) que hay actualmente en caché, más algunos bits de control (**para administración y control de acceso a caché**).

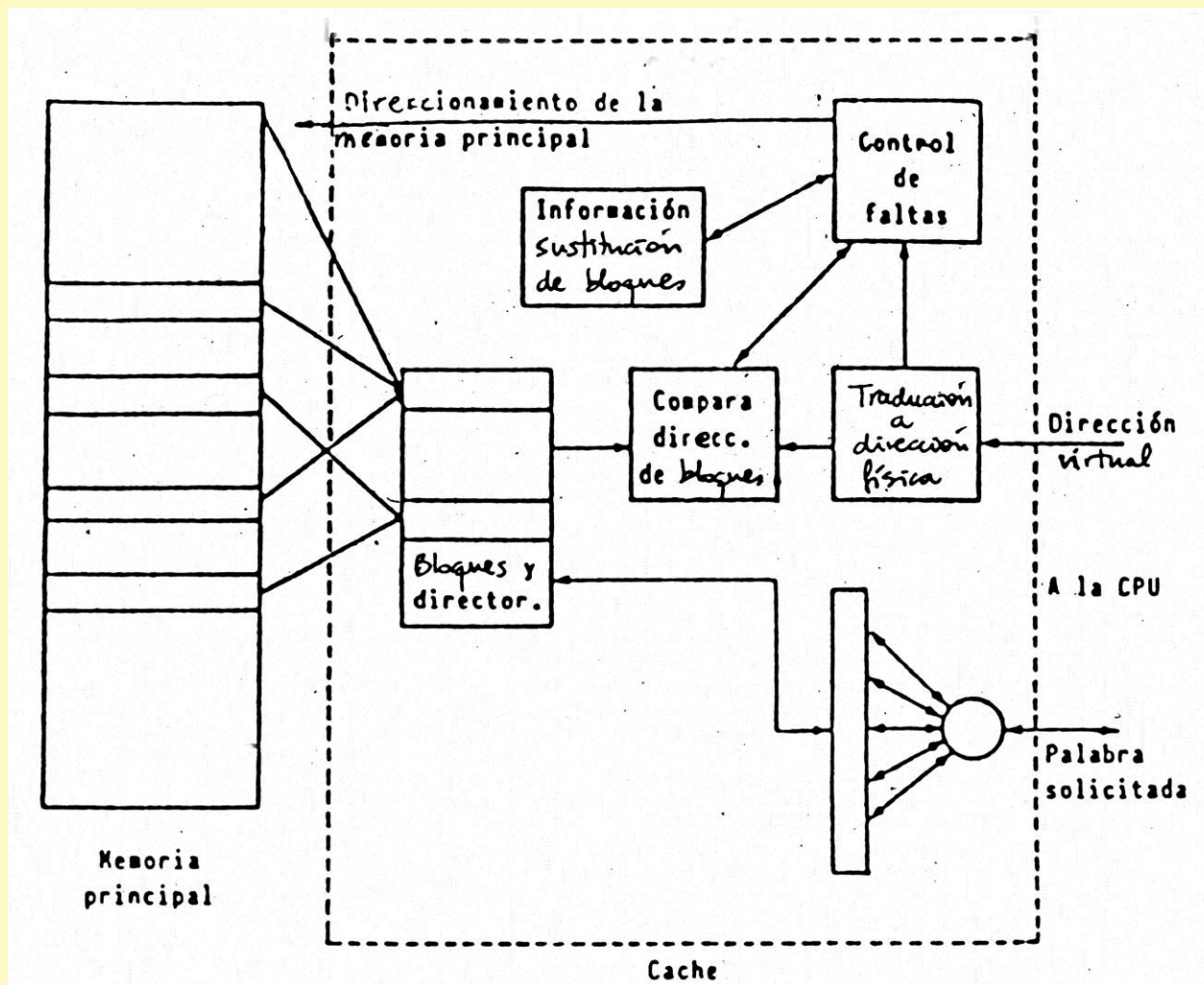
Se puede ver la caché como una colección de pares DIRECCIÓN - BLOQUE

Dirección del bloque  
en la MP

Copia del  
contenido de ese  
bloque



## OPERACIÓN DE UNA CACHÉ





## Generación de una dirección virtual por el procesador

MRT ≡ Memoria de Reserva de Traducciones (TLB)

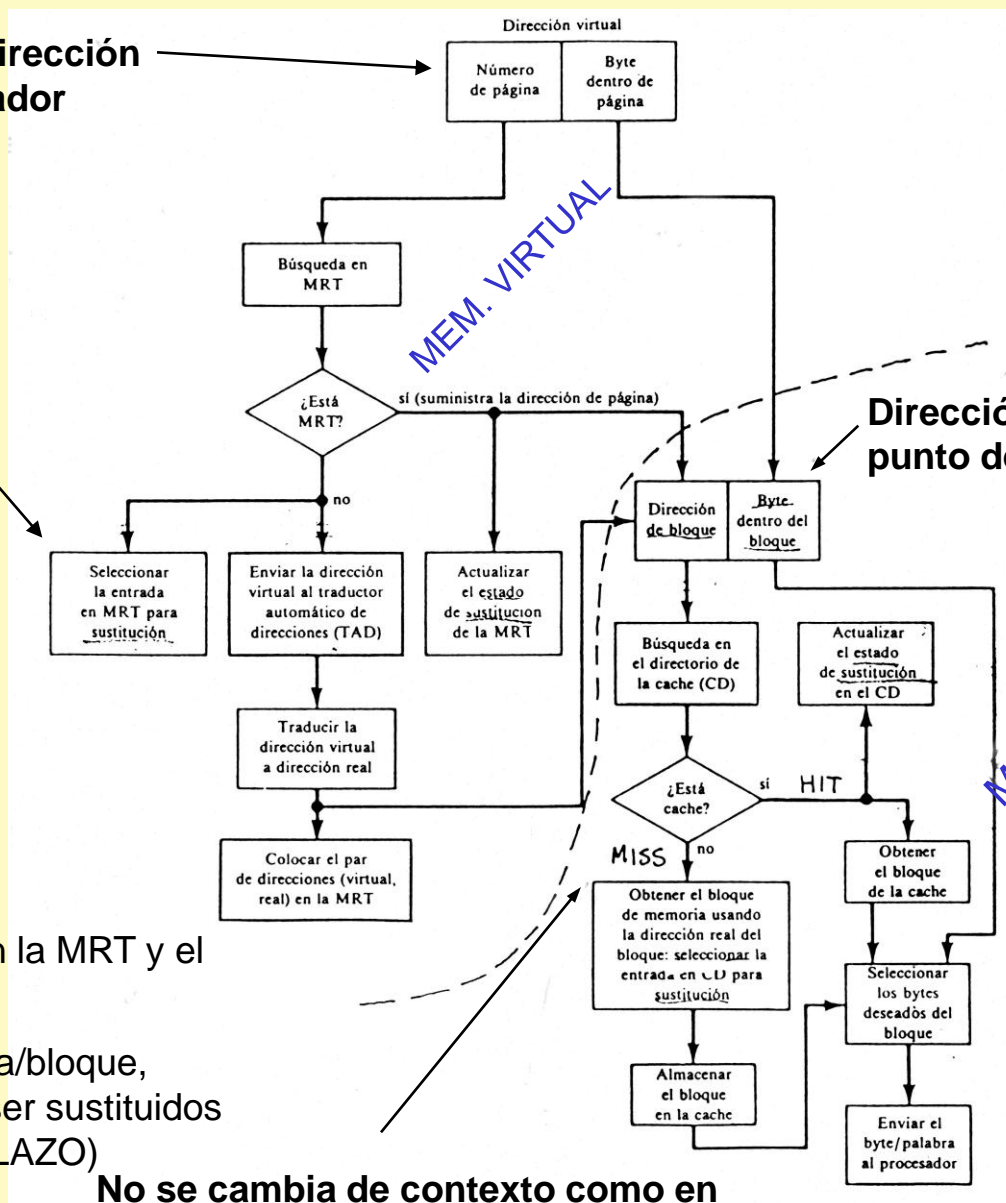
Pequeña mem.asociativa

Almacena pares dir.virtual - física

Estados de Sustitución en la MRT y el CD:

Determinan en qué página/bloque, respectivamente, deben ser sustituidos (POLÍTICAS DE REEMPLAZO)

No se cambia de contexto como en paginación.



Dirección física desde el punto de vista de la caché

MEM. CACHE

MEM. VIRTUAL



## ASPECTOS DEL DISEÑO DE UNA MEMORIA CACHÉ

Parámetros determinantes en el correcto funcionamiento y efectividad de un sistema caché:

- Política de extracción o algoritmo de búsqueda de la caché:  
*¿Cuándo y qué información va a ser transferida de la MP a la caché?*
- Política de colocación o algoritmo de ubicación de información en la caché:  
*¿Dónde se coloca en la caché la información transferida desde la MP?*
- Política o algoritmo de reemplazo:  
*Cuando se produce una falta y la caché está llena, ¿qué bloque de los que hay se sustituye por el nuevo?*
- Política o algoritmo de actualización de la MP:  
*¿Cuándo debe modificarse el contenido de la MP tras escribir en la caché?*
- Cachés de Datos / Cachés de Instrucciones.
- Tamaños del bloque y de la caché.



## POLÍTICAS DE EXTRACCIÓN O ALGORITMOS DE BÚSQUEDA

- Decide **cuándo** y **qué** información se va a introducir en la caché.
- Objetivo: **Maximización del índice de aciertos.**
- Se distinguen **dos criterios** de búsqueda básicos:
  - **Por Demanda** : Búsqueda de la información cuando se necesita.
  - **Prebúsqueda** : Búsqueda de la información antes de ser necesitada.



## BÚSQUEDA POR DEMANDA

- El algoritmo es más sencillo.
- Actúa cuando el procesador solicita un bloque y no está en la caché (*FALTA*).
- Se busca entonces el bloque en MP y se transfiere a caché.
- El procesador debe esperar hasta que la información solicitada le llegue desde MP.
- Todo sistema caché soporta este tipo de búsqueda.



## PREBÚSQUEDA O PREEXTRACCIÓN (POLÍTICAS ANTICIPATIVAS)

- Algoritmo más complejo, pero más eficiente.
- Busca el bloque que el procesador va a necesitar antes de que lo solicite, reduciendo la probabilidad de falta.
- Fiándonos de la propiedad de localidad espacial, se preextrae usualmente el bloque siguiente al solicitado por el procesador.
- Hay tres criterios o técnicas en cuanto a cuándo realizar la preextracción: ***Siempre, Por Falta y Marcada***.



## ✓ PREEXTRACCIÓN SIEMPRE

Se realiza la prebúsqueda del bloque  $i+1$  cada vez que el procesador referencia el bloque  $i$  de memoria por primera vez.

Incrementa el tráfico entre la caché y la memoria principal.

## ✓ PREEXTRACCIÓN POR FALTA

Consiste en prebuscar el bloque  $i+1$  si se referencia a  $i$  y se produce falta de bloque.

## ✓ PREEXTRACCIÓN MARCADA

Se prebusca el bloque  $i+1$  siempre que:

- Se hace referencia a  $i$  produciéndose falta de bloque.
- Se referencia por primera vez a un bloque  $i$  previamente extraído.

**PROBLEMA:** Se introducen en la caché bloques que posiblemente el procesador nunca referencia. Si la caché es pequeña, estos bloques reemplazan a otros que sí pueden necesitarse.

**POLÍTICAS DE EXTRACCIÓN SELECTIVA.** Por demanda o anticipativas si establecen diferencias entre datos que pueden pasar a la caché y datos que no.





## POLÍTICAS DE COLOCACIÓN

- La caché se diseña para que sea transparente al usuario y al programador.



NECESIDAD DE UNA FUNCIÓN QUE CONVIERTA UNA DIRECCIÓN DE MEMORIA PRINCIPAL EN UNA POSICIÓN DE CACHÉ

- La caché y la MP están divididas en unidades de igual tamaño:
  - Bloques o Líneas en la MP.
  - Marcos de Bloques o Líneas en la M. Caché.
- La política de colocación determina la función de correspondencia<sup>(1)</sup> entre bloques de MP y marcos de bloque de la caché.
- Puesto que la MP es mucho más grande que la caché, cada marco de bloque almacenará, a lo largo del tiempo, muchos bloques de la MP.
- Existen básicamente cuatro políticas de colocación:
  - CORRESPONDENCIA DIRECTA.
  - TOTALMENTE ASOCIATIVA.
  - ASOCIATIVA POR CONJUNTOS.
  - CORRESPONDENCIA POR SECTORES.

(1) También conocida como función de asignación o de mapeo



- Consideraremos dos ejemplos:

**EJEMPLO 1.-** Tamaño de caché 2K (2048) Palabras. } → 128 MARCOS DE BLOQUE  
16 Palabras por bloque. }  
Memoria principal 256K Palabras. } → → →  $2^{14} = 16384$  BLOQUES  
Dirección Física  $\equiv 18$  bits

**EJEMPLO 2.-** Tamaño de caché  $2^{2+w}$  Palabras. } → 4 MARCOS DE BLOQUE o  
 $2^w$  Palabras por bloque. } LÍNEAS DE CACHÉ  
Memoria principal  $2^{4+w}$  Palabras. } → 16 BLOQUES o LÍNEAS DE MP  
Dirección Física  $\equiv 4+w$  bits

En general supondremos que la MP tiene  $2^N$  bloques, la caché tiene  $2^n$  marcos de bloque y cada bloque tiene  $2^w$  palabras.

Una dirección física tendrá  $N+w$  bits.

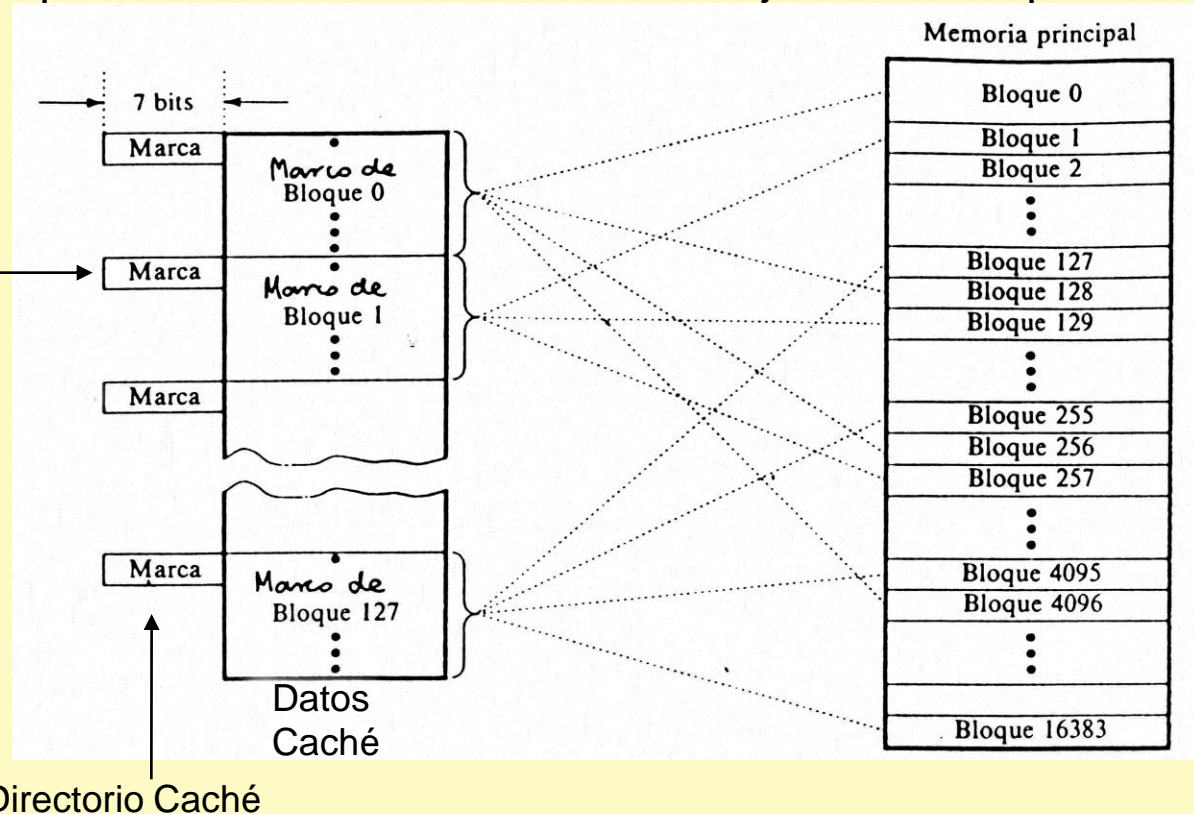
Ejemplo 1:  $N = 14$  ,  $n = 7$  ,  $w = 4$

Ejemplo 2:  $N = 4$  ,  $n = 2$



## CORRESPONDENCIA DIRECTA

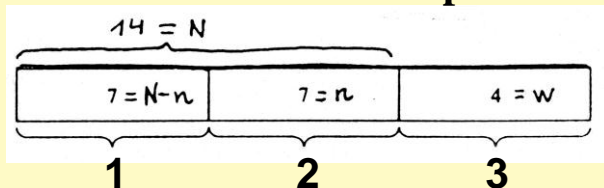
- El bloque  $i$  de la MP se corresponde con el marco de bloque  $(i \bmod 2^n)$ .
- Cada marco de bloque puede tener sólo un cierto subconjunto de bloques de MP.



Almacena los  $N-n$  bits más significativos del bloque de MP que hay en ese marco de bloque.

Sirve para identificar cuál es el bloque que hay ahí.

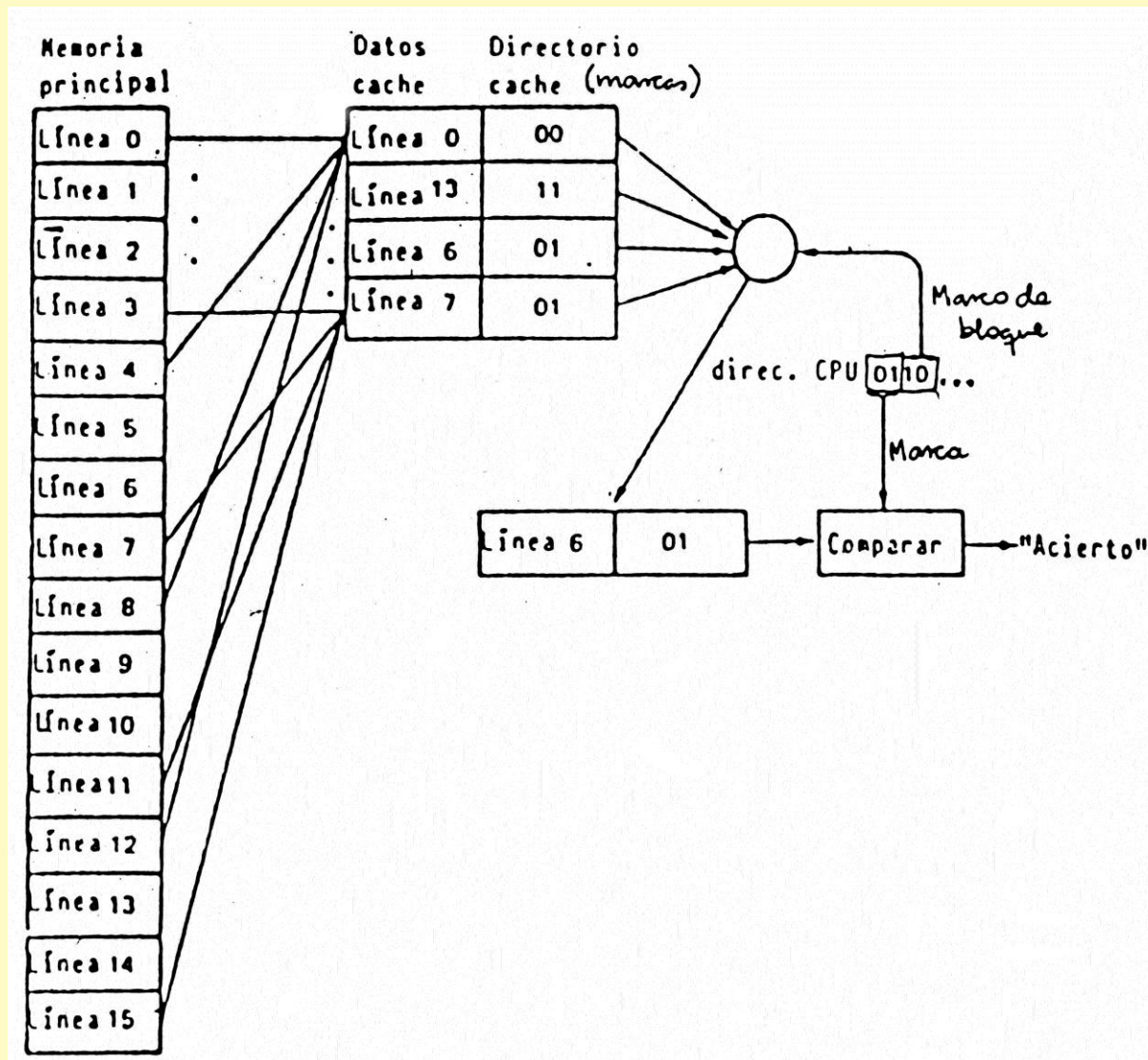
Dirección de Mem. Principal



**1. MARCA:** Identifica cada uno de los bloques asociados a ese mismo marco de bloque.

**2. MARCO DE BLOQUE,** donde debe residir ese bloque de MP

**3. PALABRA** dentro del bloque.



## VENTAJAS:

Simplicidad y bajo coste

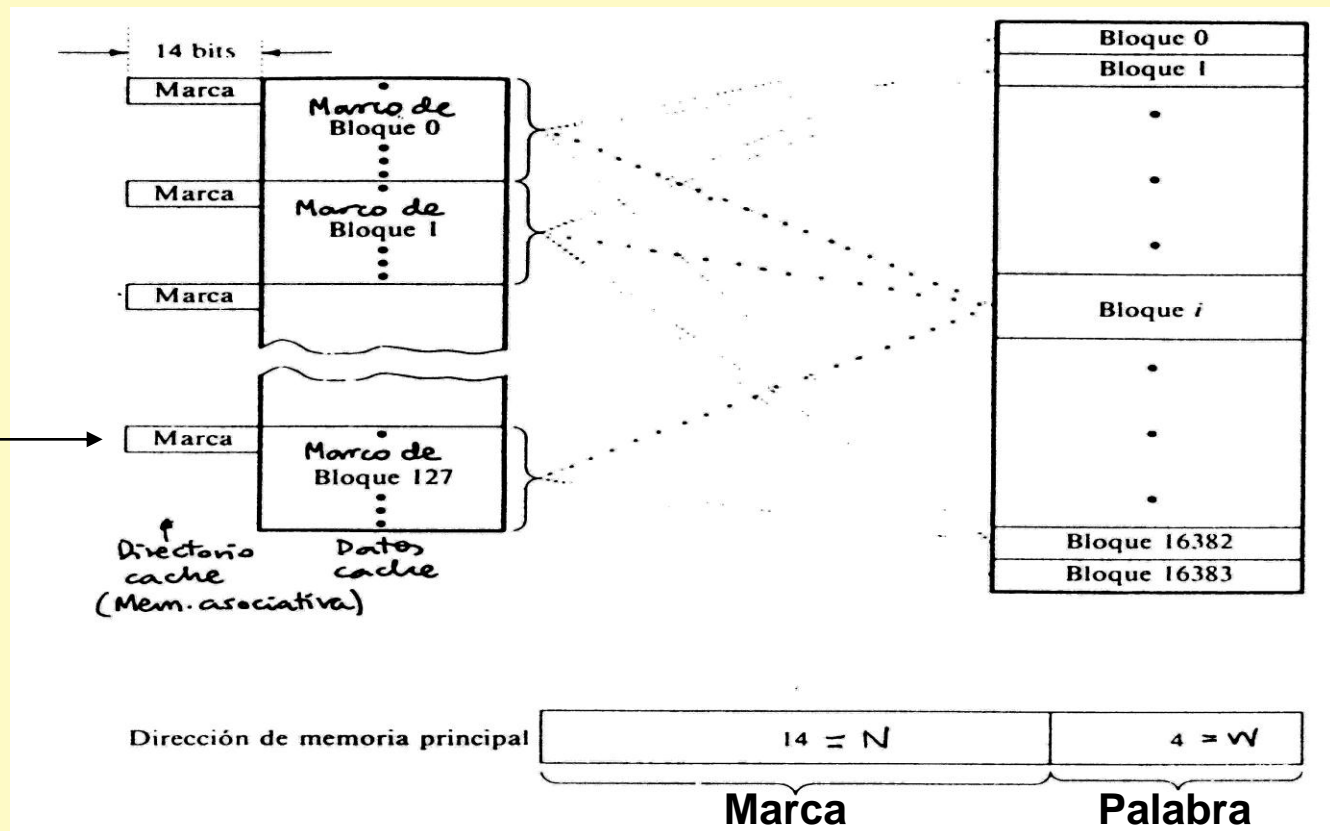
## INCONVENIENTES:

Si dos o más bloques, utilizados alternativamente, corresponden al mismo marco de bloque → el índice de aciertos se reduce drásticamente.



## CORRESPONDENCIA TOTALMENTE ASOCIATIVA (La más costosa y de mayor rendimiento)

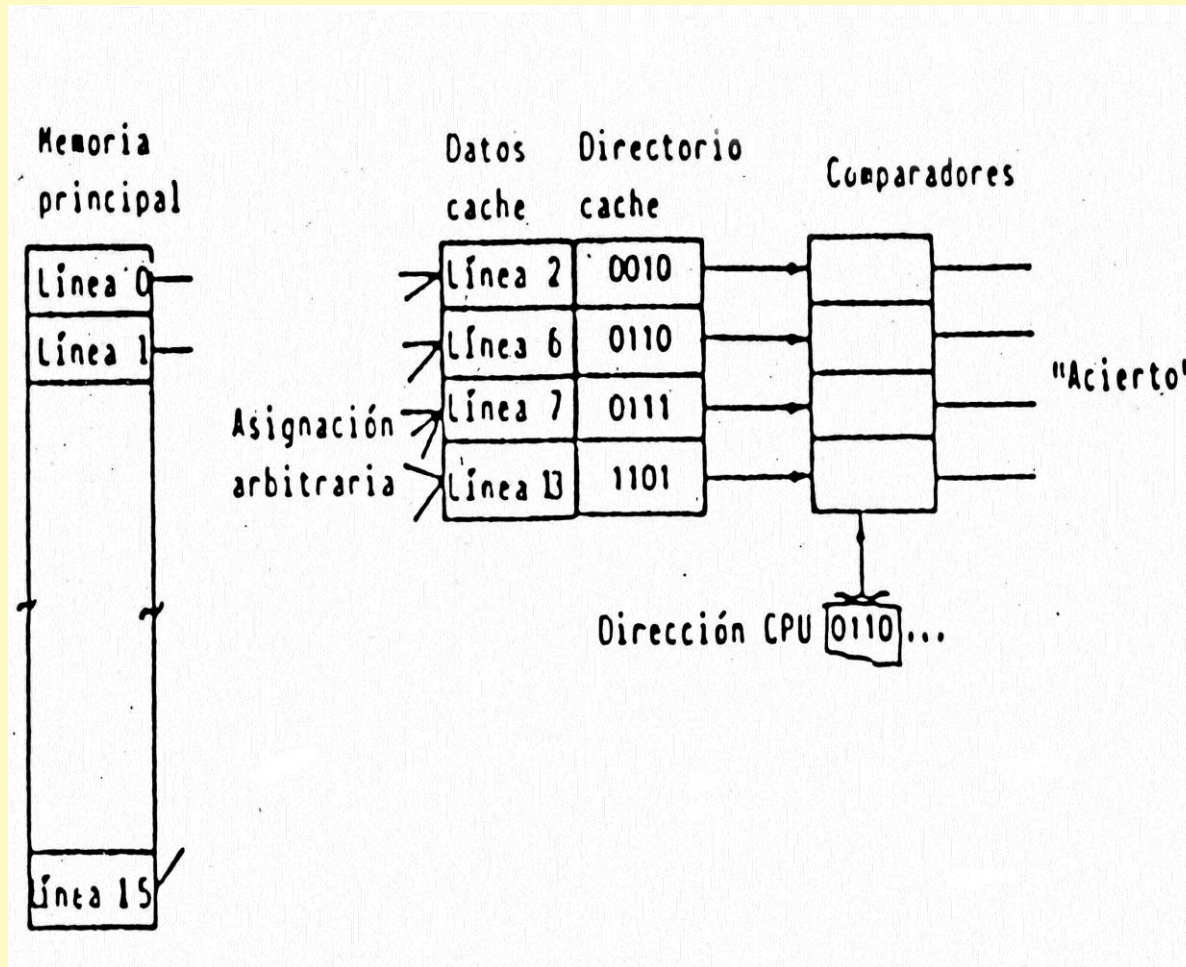
- Cualquier bloque de memoria puede estar en cualquier marco de bloque.
- Cuando se presenta una solicitud de bloque a la caché, todas las marcas se comparan simultáneamente para ver si el bloque está en caché.



Almacena los N bits del bloque de MP que hay en ese marco de bloque.

Identifica el bloque    Dentro del bloque





## VENTAJAS:

Flexibilidad permitiendo cualquier combinación de bloques de MP en la caché.



Elimina en gran medida conflictos entre bloques.

## INCONVENIENTES:

Compleja y costosa de implementar (por la memoria asociativa).

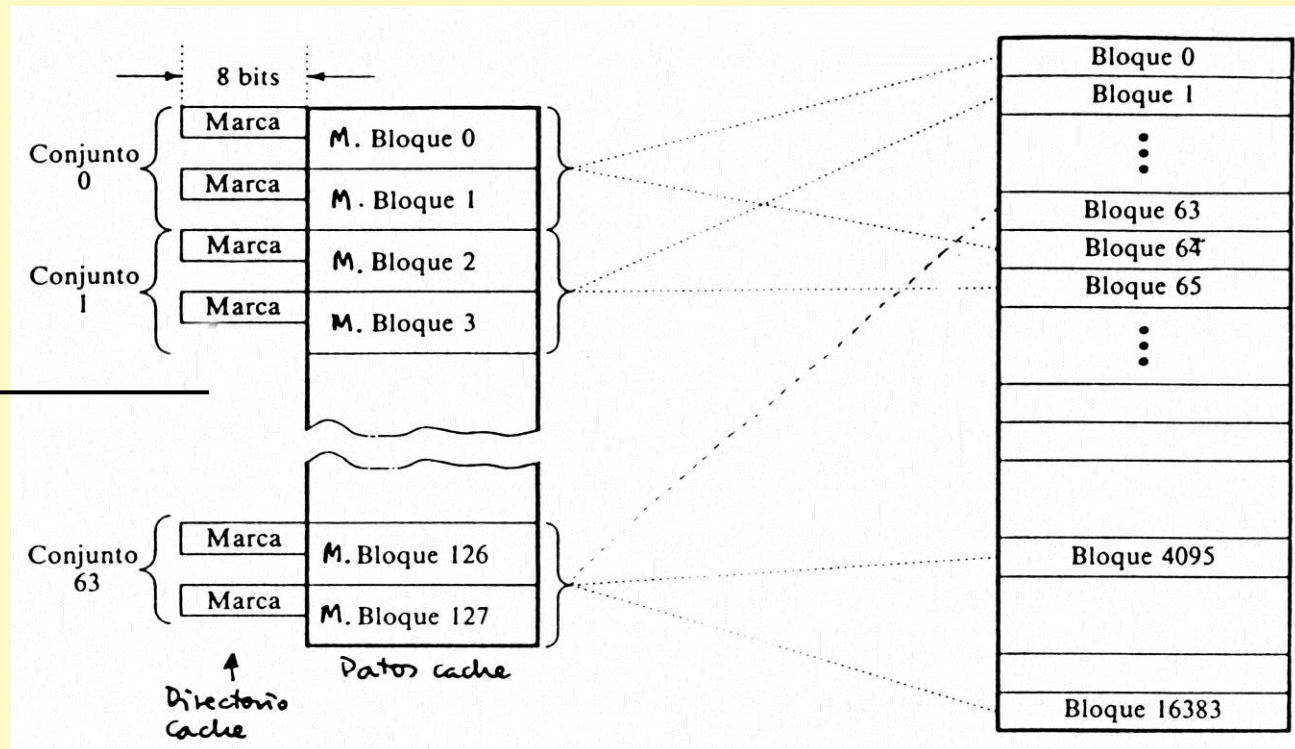


## CORRESPONDENCIA ASOCIATIVA POR CONJUNTOS (Compromiso entre directa y totalmente asociativa)

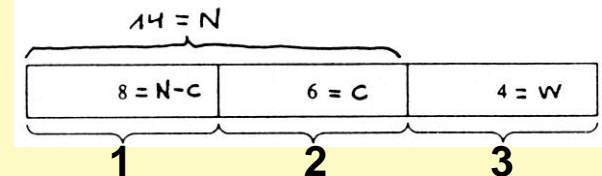
- La caché se subdivide en  $2^c$  conjuntos disjuntos  $\rightarrow 2^{n-c}$  marcos de bloque/conjunto.
- El bloque  $i$  de la MP sólo puede ir al conjunto  $(i \bmod 2^c)$ , y dentro de ese conjunto puede ir a cualquier marco de bloque.

Almacena los  $N-c$  bits más significativos del bloque de MP que hay en ese marco de bloque.

Sirve para identificar cuál es el bloque que hay ahí.



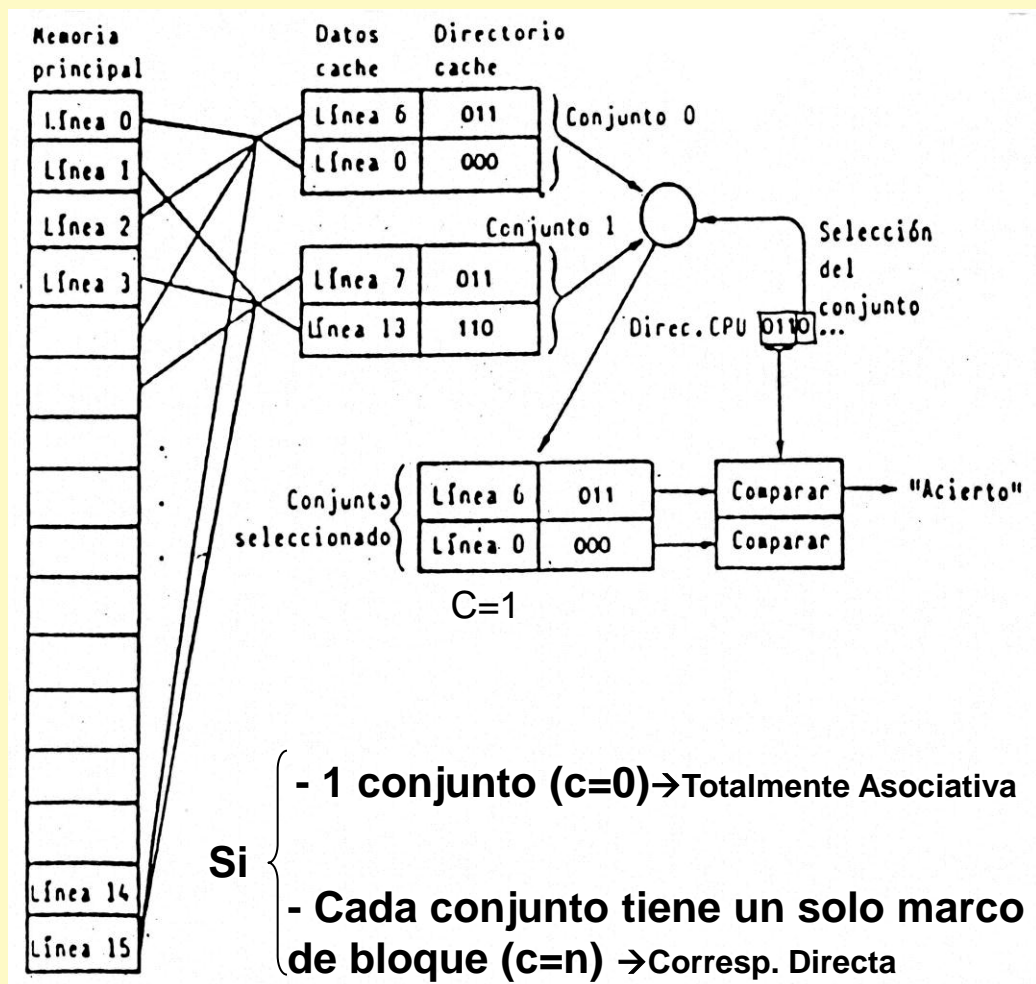
- MARCA:** Identifica cada uno de los bloques donde puede ser copiado asociado a ese mismo conjunto.
- CONJUNTO DE CACHÉ,** ese bloque.
- PALABRA** dentro del bloque.





Dos fases en el acceso a caché:

1. Selección directa del conjunto donde puede encontrarse ese bloque.
2. Búsqueda totalmente asociativa (dentro del conjunto) de la marca.



$0 < c < n \rightarrow$  Se pretende reducir el coste de la totalmente asociativa manteniendo un rendimiento similar

Es la técnica más utilizada

Resultados

experimentales demuestran que, para monoprocesadores, un tamaño de conjunto de 2 a 16 marcos de bloque funciona casi tan bien como una correspondencia totalmente asociativa con un incremento de coste pequeño respecto de la correspondencia directa.





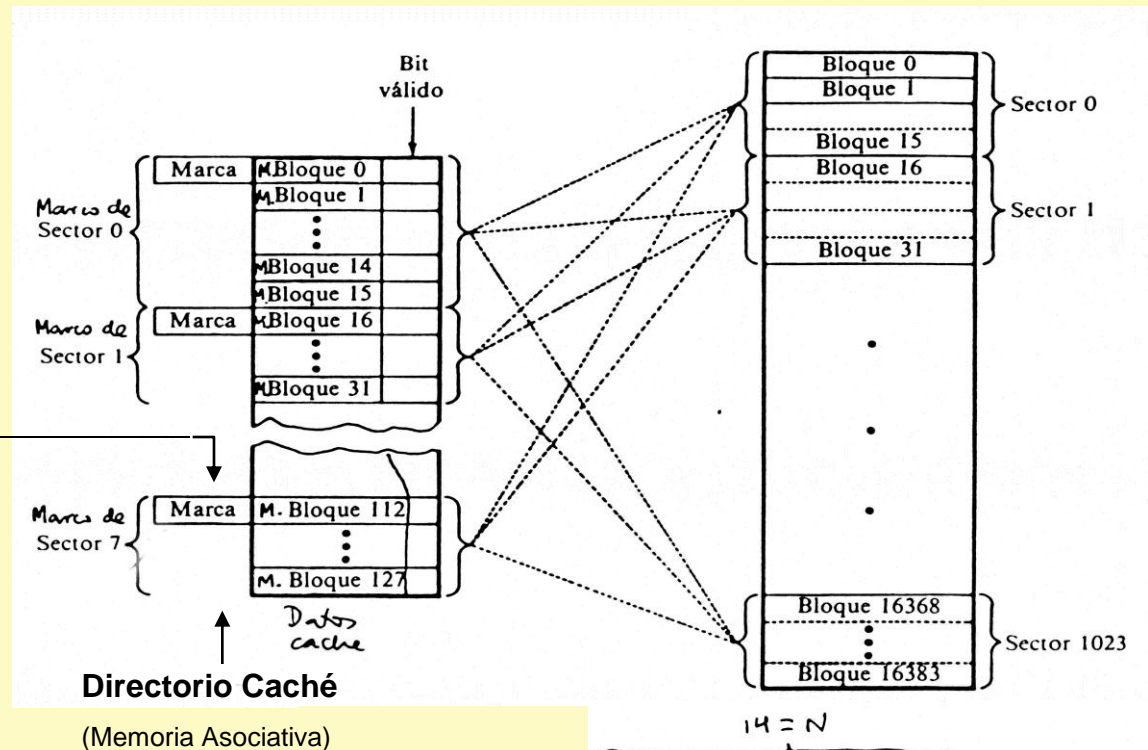
## CORRESPONDENCIA POR SECTORES

- La MP se subdivide en  $2^s$  conjuntos disjuntos llamados **SECTORES**  $\rightarrow 2^{N-s}$  bloques/sector.
- La cachés se compone de marcos de sectores, cada uno con  $2^{N-s}$  marcos de bloque.
- Un sector de MP puede estar en cualquier marco de sector, pero la correspondencia de bloques dentro de un sector es directa.

- Se lleva a caché el bloque que provocó la falta; los restantes marcos de bloque dentro de ese sector se marcan como inválidos al cargar el nuevo sector.

Almacena los  $s$  bits del sector de MP que hay en ese marco de sector.

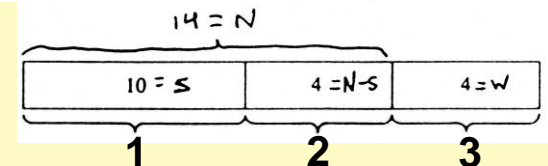
**Dentro de cada sector hay  $2^{N-s}$  bloques consecutivos.**



**1. SECTOR (marca):**  
Identifica el sector.

**2. BLOQUE dentro del sector =** marco de bloque dentro del marco de sector donde será copiado.

**3. PALABRA**  
dentro del bloque.

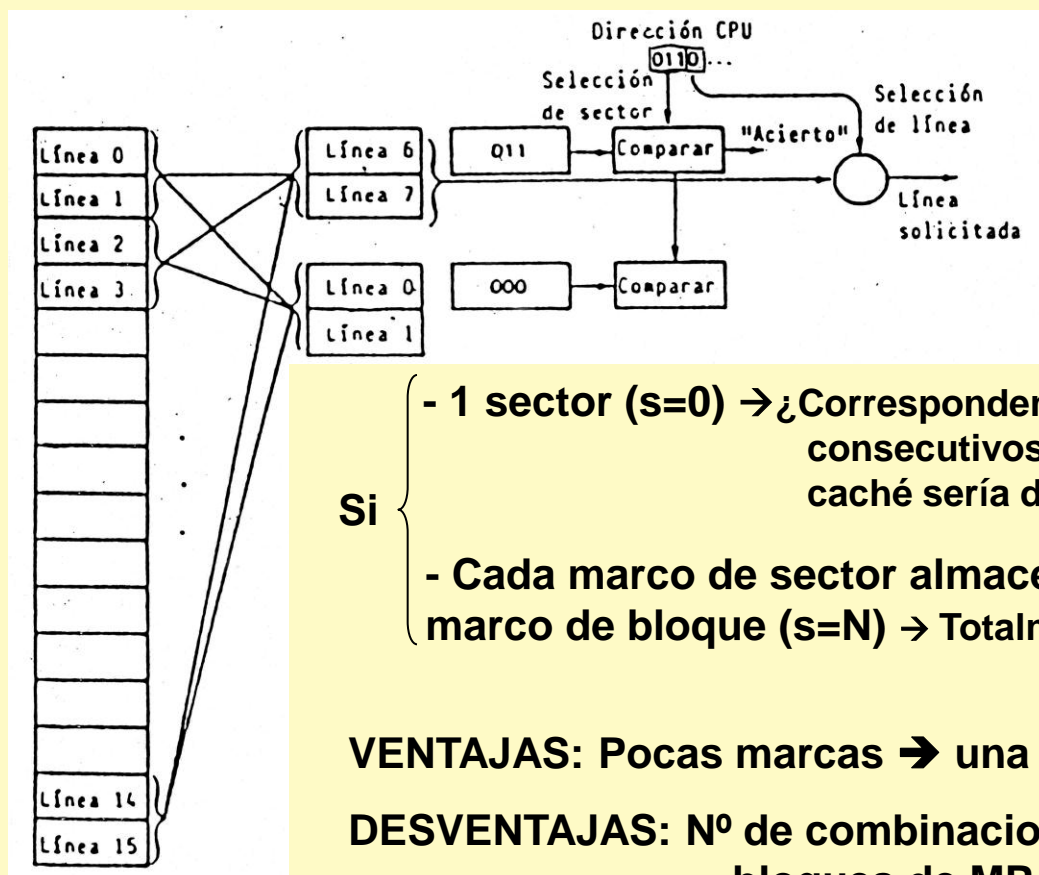


$$\text{Nº marcos de sector} = 2^{n-(N-s)}$$



Dos fases en el acceso a caché:

1. Búsqueda totalmente asociativa de la marca o sector.
2. Selección directa del marco de bloque usando los N-s bits intermedios.



- Si {
- 1 sector ( $s=0$ ) → ¿Correspondencia Directa, pero con bloques consecutivos en caché? No, porque la caché sería del tamaño de la MP
  - Cada marco de sector almacenase un solo marco de bloque ( $s=N$ ) → Totalmente Asociativa

**VENTAJAS:** Pocas marcas → una mem. Asociativa pequeña

**DESVENTAJAS:** N° de combinaciones diferentes de los bloques de MP en caché es inferior al del esquema asociativo por conjuntos.



## POLÍTICAS DE REEMPLAZO DE BLOQUES

- Si se produce una falta.
  - ↳ Hay que traer un nuevo bloque.
    - ↳ Debe decidirse, si la caché esta llena, qué bloque en caché debe ser sustituido por el nuevo.
- Este problema no existe para Correspondencia Directa.
- Para las otras organizaciones → ¿En qué posición de la caché se copia el bloque de la MP al que se accede?



## ALGORITMOS DE REEMPLAZO

Se utilizan los mismos que en memoria virtual:

- **FIFO** (First-In-First-Out): De todos los bloques existentes en la caché se reemplaza el primero que se introdujo.
- **LRU** (Least-Recently-Used): Se reemplaza el bloque menos recientemente referenciado. Es el más utilizado.
- **RAND**: El bloque a reemplazar se elige aleatoriamente de entre los de la caché.



## POLÍTICAS DE ACTUALIZACIÓN DE LA MEMORIA PRINCIPAL

- Cuando se modifica el contenido de la memoria caché debe enviarse una copia de los nuevos datos a la MP → ¿Cuándo debe realizarse esa actualización?

### ESCRITURA DIRECTA o *Escribir Siempre*

Se actualiza la MP cada vez que se modifica el contenido de la caché.

**EDAE** (Escritura Directa con Asignación en Escritura):

Se cargan bloques en memoria caché tanto si hay faltas para lectura como para escritura.

**EDSAE** (Escritura Directa Sin Asignación en Escritura):

Se cargan bloques en memoria caché si hay faltas por lectura pero no faltas por escritura.

$T(EDAE) > T(EDSAE)$  *Ya que en EDSAE no se tienen que transferir bloques a caché por faltas en escritura.*



## POST-ESCRITURA o *Escribir Bajo Falta*

Se actualiza la MP después, posiblemente cuando ese bloque debe ser reemplazado.

**PES** (Post-Escritura Siempre):

Los bloques que abandonan la caché se reescriben en MP.



**EXCESIVO TRASIEGO DE INFORMACIÓN**



**SOLUCIÓN:** Añadir un bit que indica si el bloque se ha modificado



**PEM** (Post-Escritura Marcada):

Se reescriben sólo los bloques modificados.

$\overline{T}(\text{PES}) > \overline{T}(\text{PEM})$  *Ya que en PEM no se tienen que actualizar en MP bloques no modificados.*



## CACHÉS SEPARADAS DATOS / INSTRUCCIONES

Es común particionar la memoria caché en **DOS MÓDULOS DIFERENTES**

### CACHÉ DE DATOS

- La localidad de los datos no es tan buena como la de las instrucciones → La caché de datos es menos eficiente.
- Es más compleja por la posibilidad de modificación de los datos.

### CACHÉ DE INSTRUCCIONES

- Es fácil que los lazos de los programas y pequeñas rutinas entren totalmente en caché, permitiendo su ejecución sin necesidad de acceder a la MP.
- Se simplifica la caché, ya que es habitual que se prohíba escribir en las localizaciones donde se encuentran las instrucciones → Caché sólo de lectura.



**MUCHOS SISTEMAS NO ADOPTAN CACHÉ DE DATOS**



Las cachés de instrucciones tienen menor frecuencia de fallos que las de datos.

Tamaño	Instrucción sólo	Sólo Datos	Unificada
0.25 KB	22.2 %	26.8 %	28.6 %
0.50 KB	17.9 %	20.9 %	23.9 %
1 KB	14.3 %	16.0 %	19.0 %
2 KB	11.6 %	11.8 %	14.9 %
4 KB	8.6 %	8.7 %	11.2 %
8 KB	5.8 %	6.8 %	8.3 %
16 KB	3.6 %	5.3 %	5.9 %
32 KB	2.2 %	4.0 %	4.3 %
64 KB	1.4 %	2.8 %	2.9 %
128 KB	1.0 %	2.1 %	1.9 %
256 KB	0.9 %	1.9 %	1.6 %

***Frecuencia de fallos para cachés de distintos tamaños de sólo datos, sólo instrucciones, y unificadas.*** Los datos son para una caché asociativa de 2 vías utilizando reemplazo LRU con bloques de 16 bytes para un promedio de trazas de usuario/sistema en la VAX-11 y trazas de sistema en el IBM 370 (Hill 1987). El porcentaje de referencias a instrucciones en estas trazas es aproximadamente de 53 por 100.



¿Cuál tiene la frecuencia de fallos más baja?

¿Una caché de instrucciones de 16 KB + caché de datos de 16 KB ?

¿O una caché unificada de 32 KB ?

La frecuencia global de fallos para la caché dividida es:

$$53\% * 3.6\% + 47\% * 5.3\% = \mathbf{4.4\%}$$

Una caché unificada de 32KB tiene una frecuencia de fallos ligeramente más baja: **4.3%**

Sin embargo, es común usar cachés separadas porque:

- Se pueden emitir direcciones de instrucción y dato a la vez, doblando el ancho de banda entre la caché y la CPU.
- Se puede optimizar cada caché separadamente:

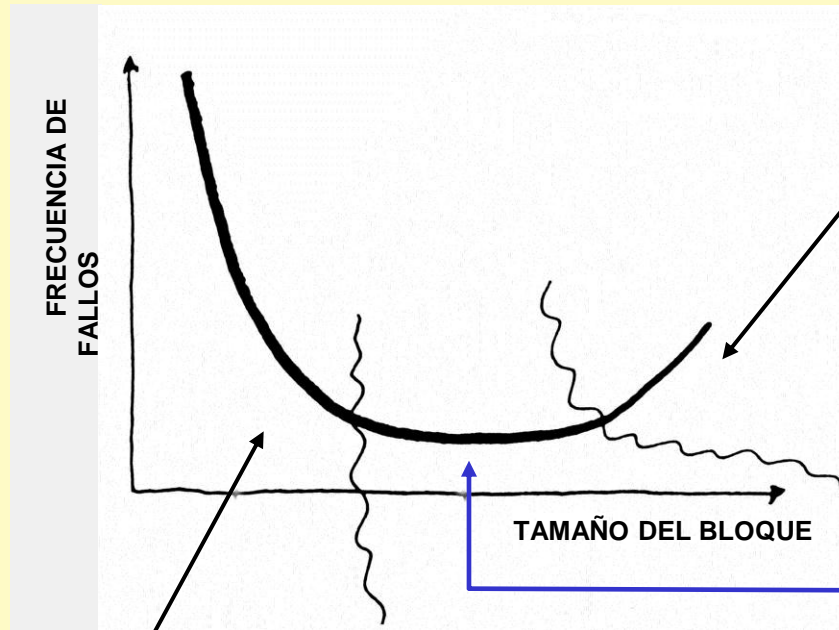
Diferentes capacidades, tamaños de bloque y asociatividades logrando un menor tiempo medio de acceso.





## TAMAÑOS DEL BLOQUE Y DE LA CACHÉ

### PARA UN TAMAÑO DE CACHÉ FIJO



Aquí la localidad espacial ya está aprovechada al máximo, pero el desaprovechamiento de la localidad temporal sigue generando faltas

**AMBOS EFECTOS DE LOCALIDAD SE COMPENSAN**

La tasa de faltas disminuye al aumentar el tamaño de bloque, pues se aprovecha mejor la localidad espacial, pero cuando aumenta el tamaño de bloque, el nº de bloques se ve disminuido → Mal aprovechamiento de la localidad temporal



Además:

Tamaño de bloque  $\uparrow\uparrow$

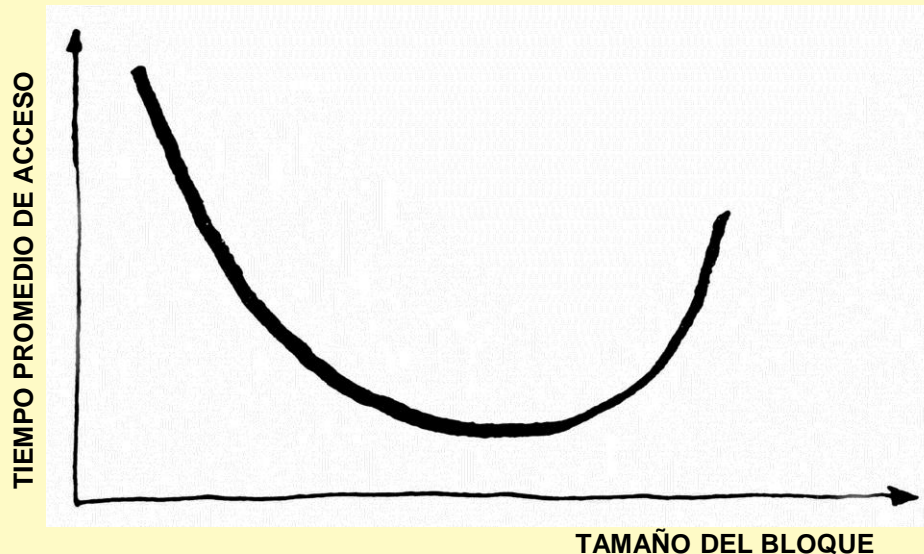


Tiempo de transferencia  $\uparrow$



Tiempo promedio de acceso  $\uparrow$  según el producto de:

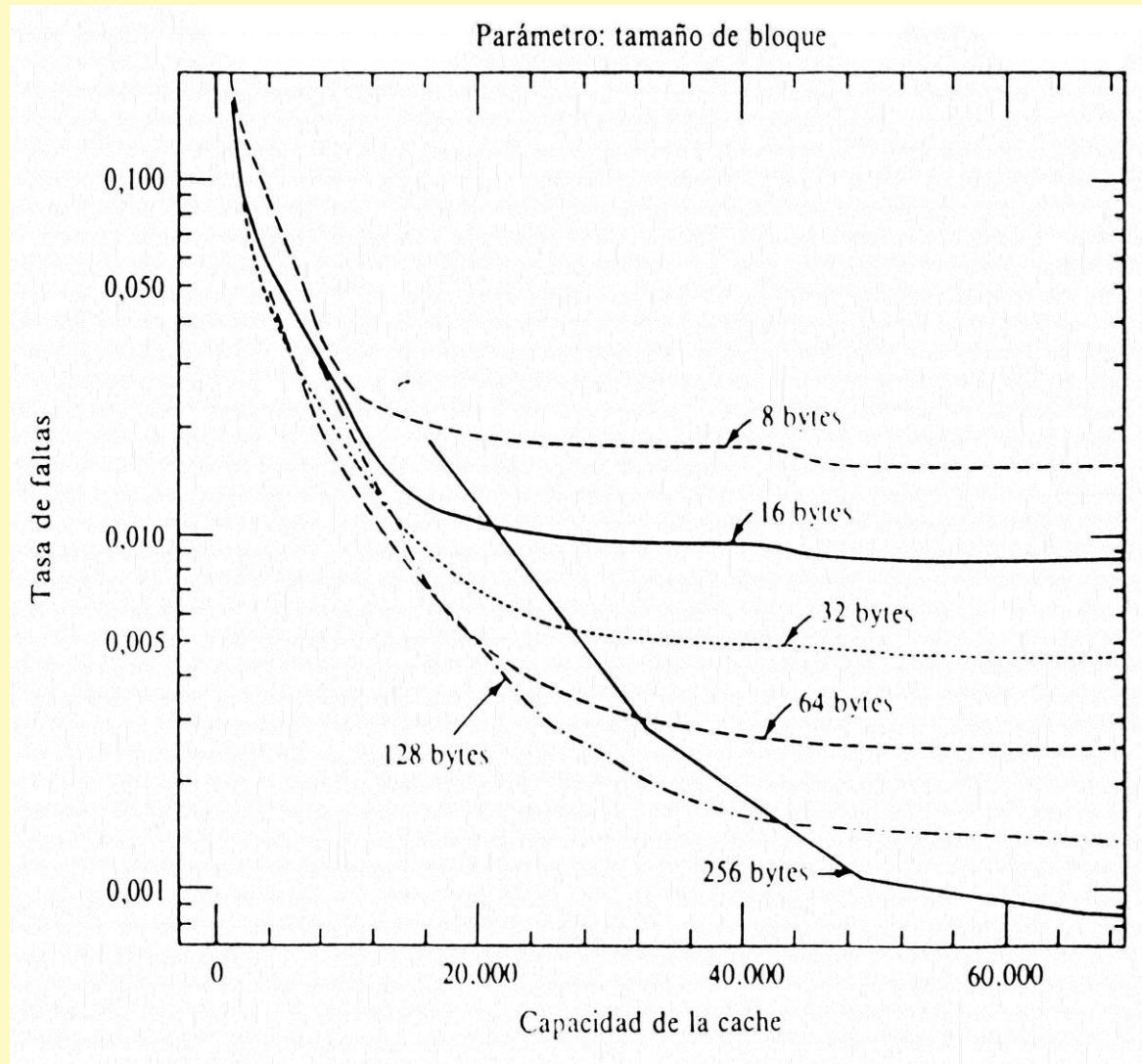
Frecuencia de fallos  $\times$  Tiempo de penalización por fallos



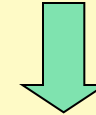


## PARA UN TAMAÑO DE BLOQUE FIJO

Un aumento del tamaño de la caché → Mayor localidad temporal contenida en la caché



la caché



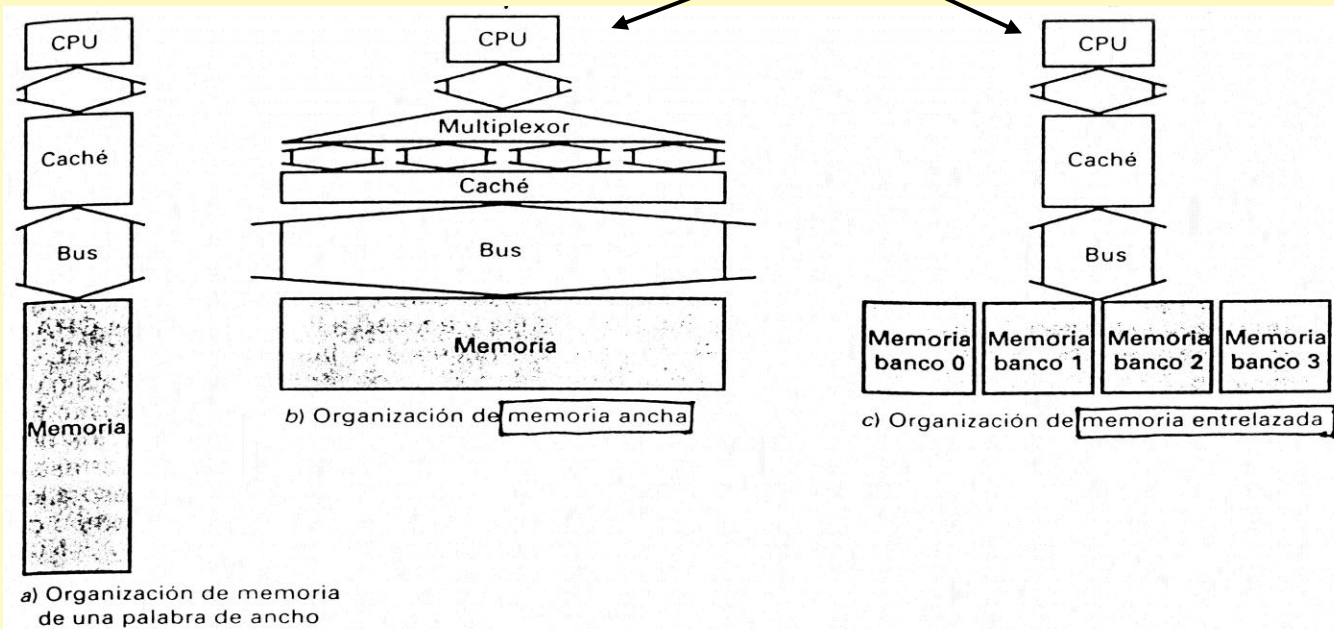
**MEJORA EN LA TASA  
DE FALTAS**





## DISEÑO DEL SISTEMA DE MEMORIA PARA SOPORTAR CACHÉS

Aunque sea difícil reducir el tiempo para buscar la primera palabra de memoria en el caso de un fallo de caché, se puede reducir la penalización de fallos incrementando el ancho de banda entre la MP y la caché.



Acceder a la MP en *modo página* aprovechando las penalizaciones de las DRAM (arrays cuadrados, acceso dividido en acceso a filas y columnas, buffer de fila).

El método principal de conseguir mayor anchura de banda de memoria es incrementar la anchura física o lógica del sistema de memoria. En esta figura hay dos formas en las que se mejora la anchura de banda de memoria. El diseño más simple, a), utiliza una memoria donde todos los componentes son de una palabra; b) muestra memoria, bus y caché de más anchura; mientras c) muestra un bus estrecho y una caché con una memoria entrelazada.



Ejemplo:  $\left\{ \begin{array}{l} 1 \text{ ciclo de reloj para enviar la dirección} \\ 10 \text{ ciclos de reloj para cada acceso iniciado a la DRAM} \\ 1 \text{ ciclo de reloj para enviar una palabra de datos} \end{array} \right.$

**TIEMPOS**

**BLOQUE DE CACHÉ: 4 PALABRAS**

- **ORGANIZACIÓN DE UNA PALABRA DE ANCHO**  
**Penalización de fallos=  $1 + 4 \cdot 10 + 4 \cdot 1 = 45$  ciclos de reloj**
- **ORGANIZACIÓN DE MEMORIA ANCHA: 2 PALABRAS**  
**Penalización de fallos=  $1 + 2 \cdot 10 + 2 \cdot 1 = 23$  ciclos de reloj**
- **ORGANIZACIÓN DE MEMORIA ANCHA: 4 PALABRAS**  
**Penalización de fallos=  $1 + 1 \cdot 10 + 1 \cdot 1 = 12$  ciclos de reloj**
- **ORGANIZACIÓN DE MEMORIA ENTRELAZADA**  
**Penalización de fallos=  $1 + 1 \cdot 10 + 4 \cdot 1 = 15$  ciclos de reloj**



## EJEMPLO: UNIDAD DE CACHÉ INTERNA DEL 486

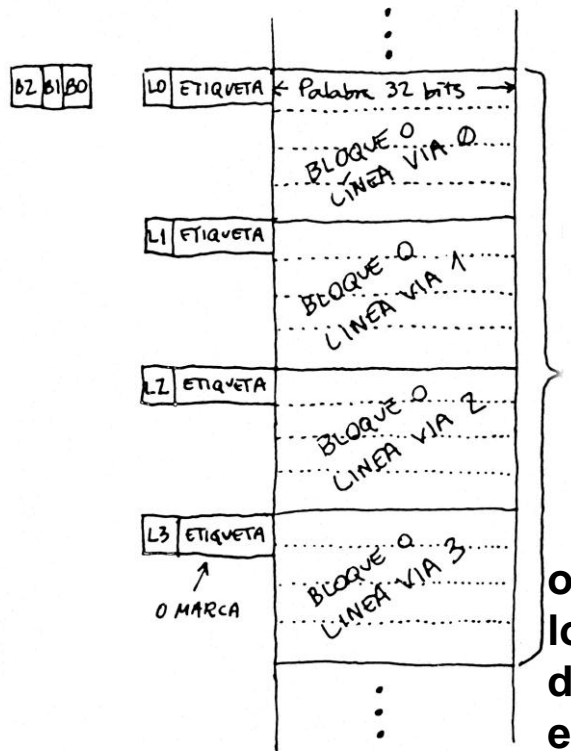
Almacena datos e instrucciones. Su uso es transparente para el software.

Estructura: Memoria caché con correspondencia asociativa por conjuntos de 8KB.

Cada conjunto tiene 4 bloques o líneas → Se llama también “ASOCIATIVA DE 4 VÍAS”.

Cada bloque tiene 4 palabras de 32 bits (2 dobles palabras o 16 bytes).

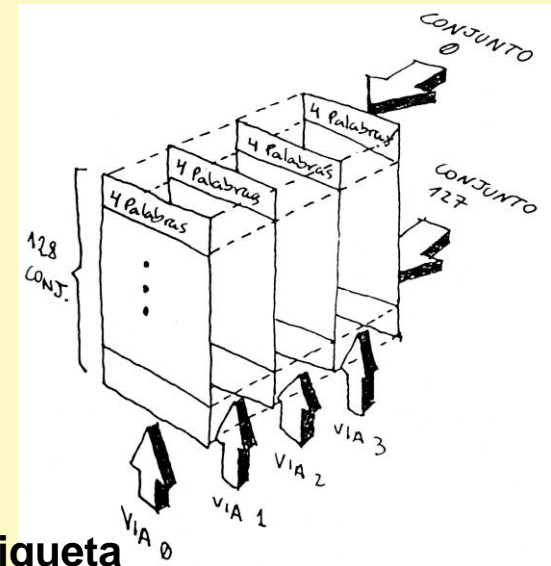
La caché tiene 128 conjuntos.



CONJUNTO

16 PALABRAS 32 BITS = 64 BYTES

Cada bloque tiene una etiqueta o marca de 21 bits, que se compara con los 21 bits de mayor peso (A<sub>11</sub>- A<sub>31</sub>) de la dirección física referenciada para ver si está en caché.



Por cada conjunto hay 7 bits de estado: B2 B1 B0 L3 L2 L1 L0

Utilizados por el algoritmo de reemplazo LRU

Indican qué líneas son válidas (contienen datos correctos)