

Memoria

Estructura de Computadores
14^a y 15^a semanas

Bibliografía:

- | | |
|--------------------|---|
| [HAM03] Cap.5 | Organización de Computadores. Hamacher, Vranesic, Zaki. McGraw-Hill 2003
Signatura ESIIT/ C.1 HAM org |
| [STA8] Caps. 4 y 5 | Organización y Arquitectura de Computadores, 7 ^a Ed. Stallings. Pearson Educación, 2008.
Signatura ESIIT/ C.1 STA org |

Guía de trabajo autónomo (4h/s)

■ Lectura

- Cap. 5 Hamacher
- Caps. 4 y 5 Stallings

Bibliografía:

[HAM03] Cap.5

Organización de Computadores. Hamacher, Vranesic, Zaki. McGraw-Hill 2003

Signatura ESIIT/[C.1 HAM org](#)

[STA8] Caps.4 y 5

Organización y Arquitectura de Computadores, 7^a Ed. Stallings. Pearson Educación, 2008.

Signatura ESIIT/[C.1 STA org](#)

[

Memoria

- **Jerarquía de memoria. Concepto de localidad**
- **Memorias RAM semiconductoras. Memorias de sólo lectura. Prestaciones: velocidad, tamaño y coste**
- **Configuración y diseño de memorias utilizando varios chips**
- **Memorias asociativas**
- **Memoria cache. Influencia en las prestaciones**

Introducción

- **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 memoria
- **Algunos objetivos a tener en cuenta en el diseño del sistema de memoria:**
 - Capacidad de almacenamiento y velocidad de acceso suficientes.
 - Coste por bit reducido.
 - Liberar a los programadores de realizar tareas de gestión de memoria.

Introducción

- Consideraremos la memoria en un sentido amplio, englobando:
 - Memoria interna: registros.
 - No hay diferencia de velocidad con la CPU.
 - Memoria caché.
 - Memoria principal.
 - La CPU no puede ejecutar directamente programas que estén más allá de la memoria principal.
 - Memoria secundaria o masiva.
 - Más lenta. Discos y cintas magnéticas, CD-ROM, etc.
- En este sentido amplio, llamaremos tiempo de acceso al tiempo que se requiere para leer (o escribir) un dato (palabra) en la memoria.

Introducción

■ Según el método de acceso las memorias se pueden clasificar en tres grupos:

- De acceso **aleatorio** (RAM). ————— **Random Access Memory**
 - El tiempo de acceso es independiente de la posición de memoria a acceder. Ej.: SRAM, DRAM, ROM, etc.
- De acceso **secuencial** (SAM). ————— **Sequential Access Memory**
 - El tiempo de acceso depende de la posición de los datos. Ej.: Cinta magnética.
- De acceso **semialeatorio o directo** (DASD). ————— **Direct Access Storage Device**
 - Principalmente discos, en los que se accede a las pistas de forma aleatoria y a los datos en las pistas de forma secuencial.

Téngase en cuenta que en SAM y DASD el tiempo de acceso a un bloque de n palabras no es $n \cdot$ Tiempo de acceso, sino:

Tiempo de acceso (a la primera palabra) + Tiempo de bloque

Aumento del ancho de banda

■ Ancho de banda de memoria de un ordenador:

- Se define como:

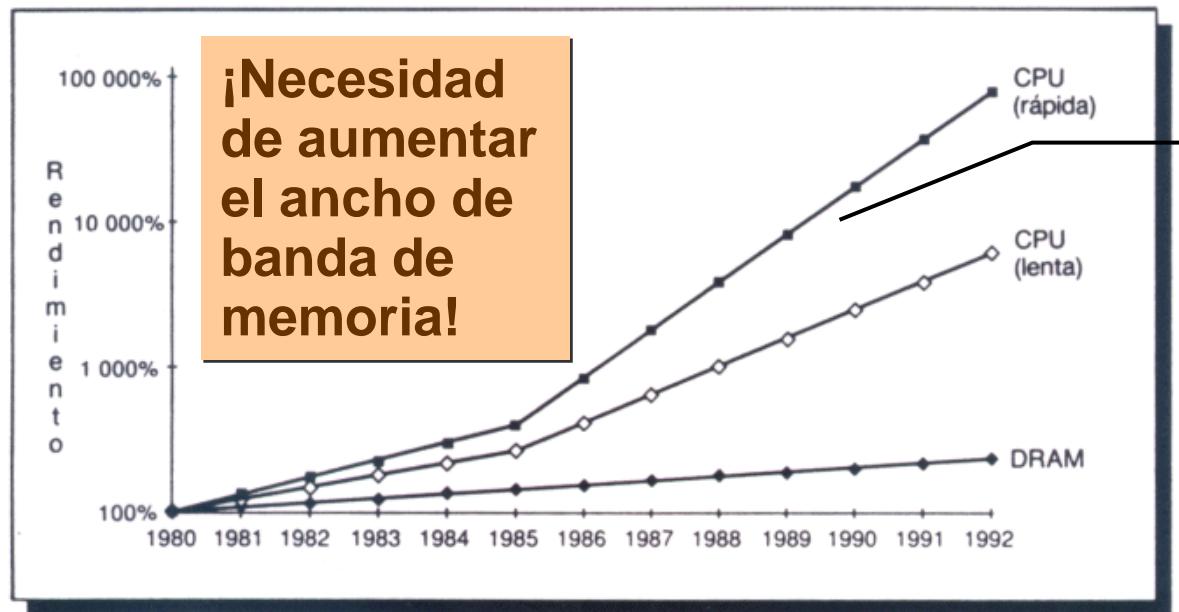
Número de palabras a las que puede acceder el procesador (o que se pueden transferir entre el procesador y la memoria) por unidad de tiempo

■ Problema:

- Diferencia entre el rendimiento del procesador y el ancho de banda de memoria (“processor-memory gap”).

Aumento del ancho de banda

- Diferencia de rendimiento entre CPU y memoria:



Comenzando con el rendimiento de 1980 como punto de partida, se dibuja el rendimiento de las DRAM y CPU a lo largo del tiempo.

El punto de partida de DRAM de 64 KB en 1980, con tres años para la siguiente generación. La línea de CPU lentas supone una mejora del 19 por 100 por año hasta 1985 y una mejora del 50 por 100 después. La línea de CPU rápidas supone una mejora del rendimiento del 25 por 100 entre 1980 y 1985 y un 100 por 100 por año después. Observar que el eje vertical debe estar en la escala logarítmica para registrar el tamaño del salto de rendimientos CPU-DRAM.

Ley de Moore:
“Las prestaciones del procesador se duplican cada 18 meses.”

Aumento del ancho de banda

■ Diferencia de rendimiento entre CPU y memoria:

- Ejemplo (1994):
 - DEC Alpha 21164
 - 300 MHz
 - 4 instrucciones por ciclo
 - ⇒ 1200 MIPS (pico)
 - ⇒ Una instr. cada 0,833 ns
- Memoria DRAM 64 Mbits
 - Tiempo de acceso = 60 ns. ¡72 veces mayor!

Aumento del ancho de banda

■ Diferencia de rendimiento entre CPU y memoria:

- Ejemplo (2003):
 - AMD Athlon XP 2800+
 - 2,255 GHz
 - 9 instrucciones por ciclo
 - } **⇒ >20 GIPS (pico)**
 - } **⇒ Una instr. cada 0,049 ns**
 - Memoria DDR 333 MTransferencias/s *
- Una lectura/escritura cada 3 ns. **¡61 veces mayor!**
- (*) Téngase en cuenta que la DDR no es una DRAM sencilla, sino que utiliza mejoras de arquitectura, como el entrelazado.

Aumento del ancho de banda

■ Diferencia de rendimiento entre CPU y memoria:

■ Ejemplo (2009):

- Intel Core 2 Quad Q9650
 - 3 GHz
 - 4 núcleos
 - 4 instrucciones por ciclo / núcleo
 - Memoria DDR3 1333 MTransferencias/s *
 - Una lectura/escritura cada 0,75 ns. **¡36 veces mayor!**
 - (*) Téngase en cuenta que la DDR3 no es una DRAM sencilla, sino que utiliza mejoras de arquitectura, como el entrelazado.
- } **⇒ 48 GIPS (pico)**
} **⇒ Una instr. cada 0,02083 ns**

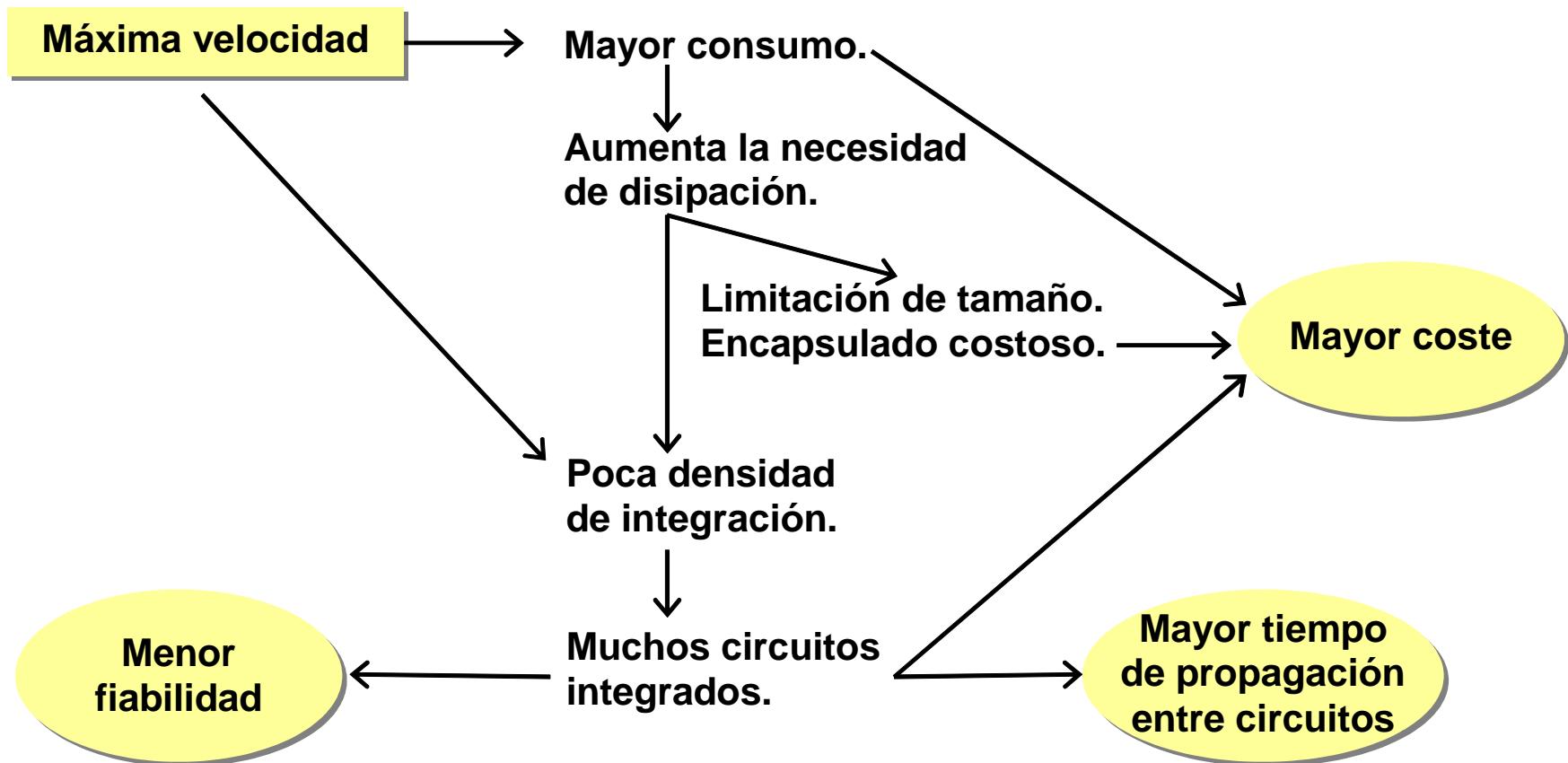
Aumento del ancho de banda

■ Métodos para aumentar el ancho de banda de memoria:

- Usar tecnología de alta velocidad.
- ✓ Organizar la memoria jerárquicamente, incluyendo memoria caché.
- ✓ Incrementar el ancho de la memoria (número de palabras accedidas simultáneamente).
- ✓ Dividir la memoria principal en un conjunto de módulos direccionables simultáneamente (memoria entrelazada).
- ✓ Emplear memorias asociativas (direcciónables por contenido) cuando sea posible.

Aumento del ancho de banda

- Disponer de memoria de la máxima capacidad con el menor tiempo de acceso y sin mejoras en la arquitectura no es sólo muy caro, sino ¡inviable!



Aumento del ancho de banda

■ Ejemplos de 1995 (memorias SRAM y DRAM clásicas sin mejoras de arquitectura):

Para cada fabricante se eligió el circuito más rápido de entre varios disponibles

(*) Demasiados chips. Ocupan mucho espacio y el retardo entre circuitos es mayor que el tiempo de acceso

Fabricante	Fecha	Memoria	Tecnolog.	T. acc.	Consumo	Nº chips	Consumo	Precio aprox.(1995)
MHS	1994	SRAM 16Kx4-Bit	BiCMOS	8 ns	Activo: 700 mW Standby: 250 mW	128 / 1 MB 2048 / 16 MB !! (*) 8192 / 64 MB !! (*)	32-89.6 W / 1 MB 512-1433 W / 16 MB !! 2048-5734.4 W / 64 MB !!	250.000 pts / 1 MB !! 4.000.000 pts / 16 MB !! 16.000.000 pts / 64 MB !!
Paradigm	1991	SRAM 32Kx8-Bit	CMOS	10 ns	Activo: 400 mW Standby: 150 mW	32 / 1MB 512 / 16MB !! (*) 2048 / 64 MB !! (*)	4.8-12.8 W / 1MB 76.8-204.8 W / 16 MB 307.2-819.2 W / 64 MB !!	65.000 pts / 1MB 1.000.000 pts / 16 MB !! 4.000.000 pts / 64 MB !!
Mosel-Vitelic	1993	DRAM 1MBx8-Bit (SIMM de 8 chips 1Mx1-Bit)	CMOS	60 ns	Activo: 3.6 W Standby: 80 mW	8 (1 SIMM) / 1MB 128 (16 SIMMs) / 16MB 512 (64 SIMMs) / 64 MB !! (*)	0.08-3.6 W / 1 MB 1.28-57.6 W / 16 MB 5.12-230.4 W / 64 MB	5.000 pts / 1 MB 80.000 pts / 16 MB 320.000 pts / 64 MB
Mosel-Vitelic	1993	DRAM 4MBx8-Bit (SIMM de 8 chips 4Mx1-Bit)	CMOS	70 ns	Activo: 3.6 W Standby: 80 mW	32 (4 SIMMs) / 16 MB 128 (16 SIMMs) / 64 MB	0.32-14.4 W / 16 MB 1.28-57.6 W / 64 MB	80.000 pts / 16 MB 320.000 pts / 64 MB
Mosel-Vitelic	1993	DRAM 4MBx32-Bit (SIMM de 32 chips 4Mx1-Bit)	CMOS	70 ns	Activo: 17.01 W Standby: 325 mW	32 (1 SIMM) / 16 MB 128 (4 SIMMs) / 64 MB	0.325-17.01 W / 16 MB 1.3-68.05 W / 64 MB	80.000 pts / 16 MB 320.000 pts / 64 MB

Localidad de las referencias

■ Aprovecha los siguientes hechos:

▪ 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.

▪ Localidad de las referencias

- Espacial
- Secuencial
- Temporal

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

Localidad de las referencias

■ Localidad de las referencias:

- Localidad **espacial** $d(t+n) = d(t) + k$, con n y k pequeños.

- “Si se referencia un elemento, los elementos cercanos a él serán referenciados pronto.”
 - Justificación: Los datos relacionados se almacenan juntos.

- Localidad **secuencial** $d(t+1) = d(t) + 1$

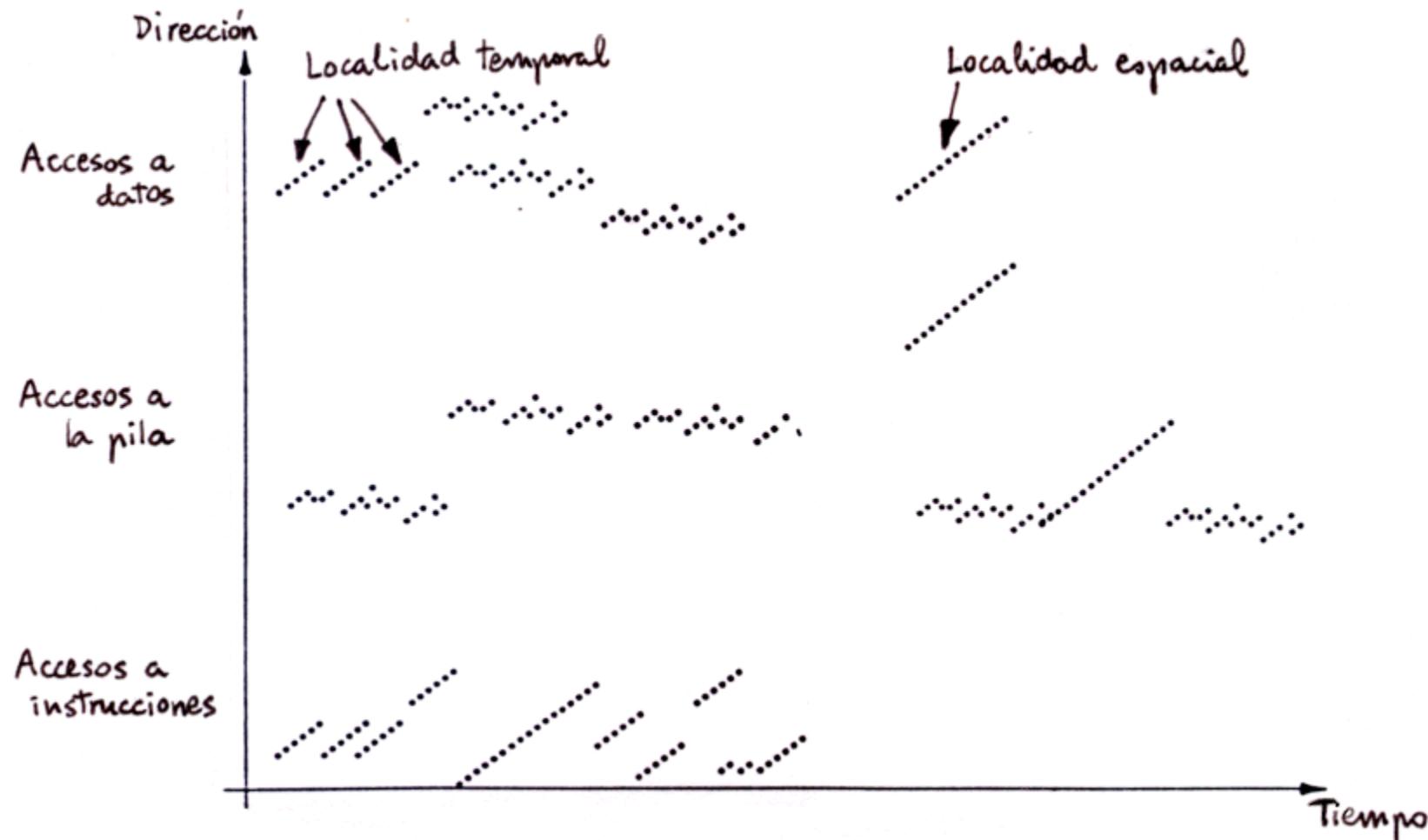
- “Las direcciones de memoria utilizadas suelen ser contiguas.”
 - Justificación: Las instrucciones se ejecutan secuencialmente.

- Localidad **temporal** $d(t+n) = d(t)$, con n pequeño.

- “Si se referencia un elemento, volverá a referenciarse 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.

Localidad de las referencias

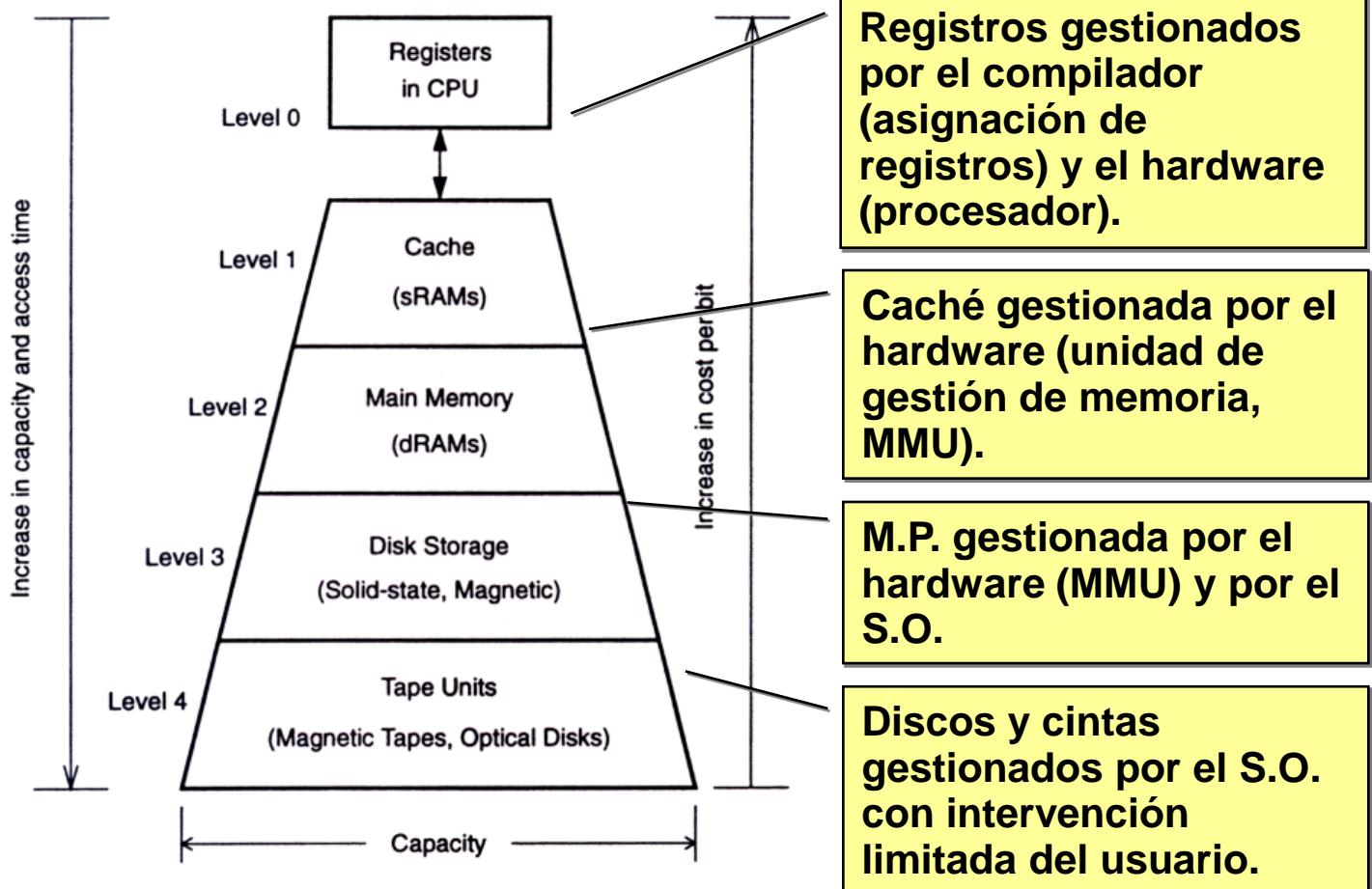
■ Patrones de referencia a memoria típicos:



Jerarquía de memoria

- 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 una capacidad menor pero es más rápido que los inferiores.
- Toda la información de un nivel se encuentra almacenada también en el siguiente.



Jerarquía de memoria

■ Parámetros que caracterizan cada nivel i

- Los dispositivos de almacenamiento (registros, memorias, discos y unidades de cinta), se caracterizan por:
 - **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 bit o 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 información que se transfiere entre el nivel i y el $i+1$.

Jerarquía de memoria

- Características de los distintos niveles de memoria en un ordenador tipo “mainframe” de 1993:

Nivel de memoria Características	Nivel 0 Registros CPU	Nivel 1 Cache	Nivel 2 Memoria principal	Nivel 3 Almac. en disco	Nivel 4 Almac. en cinta
Tecnología del dispositivo	ECL	SRAM 256 Kbit	DRAM 4 Mbit	Unidad de disco magnético de 1 Gbyte	Unidad de cinta magnética de 5 Gbyte
Tiempo de acceso t_i	10 ns	25-40 ns	60-100 ns	12-20 ms	2-20 min (tiempo de búsqueda)
Capacidad en bytes si	512 bytes	128 Kbytes	512 Mbytes	60-228 Gbytes	512 Gbytes-2 Tbytes
Coste c_i en centavos/KB	18000	72	5,6	0,23	0,01
Coste c_i en €/KB (1 \$ = 0,8247 €)	148,45 €	59,38 cents.	4,62 cents.	0,19 cents.	0,01 cents.
Coste total en €	74,22 €	76,00 €	24.213,30 €	119.336,97 €	44.275,74 €
Ancho de banda b_i (en MB/s)	400-800	250-400	80-133	3-5	0,18-0,23
Unidad de transferencia x_i	4-8 bytes por palabra	32 bytes por bloque	0,5-1 Kbytes por página	5-512 Kbytes por fichero	Almacenamiento de seguridad

Se verifica que:

$$t_i < t_{i+1}$$

$$s_i < s_{i+1}$$

$$c_i > c_{i+1}$$

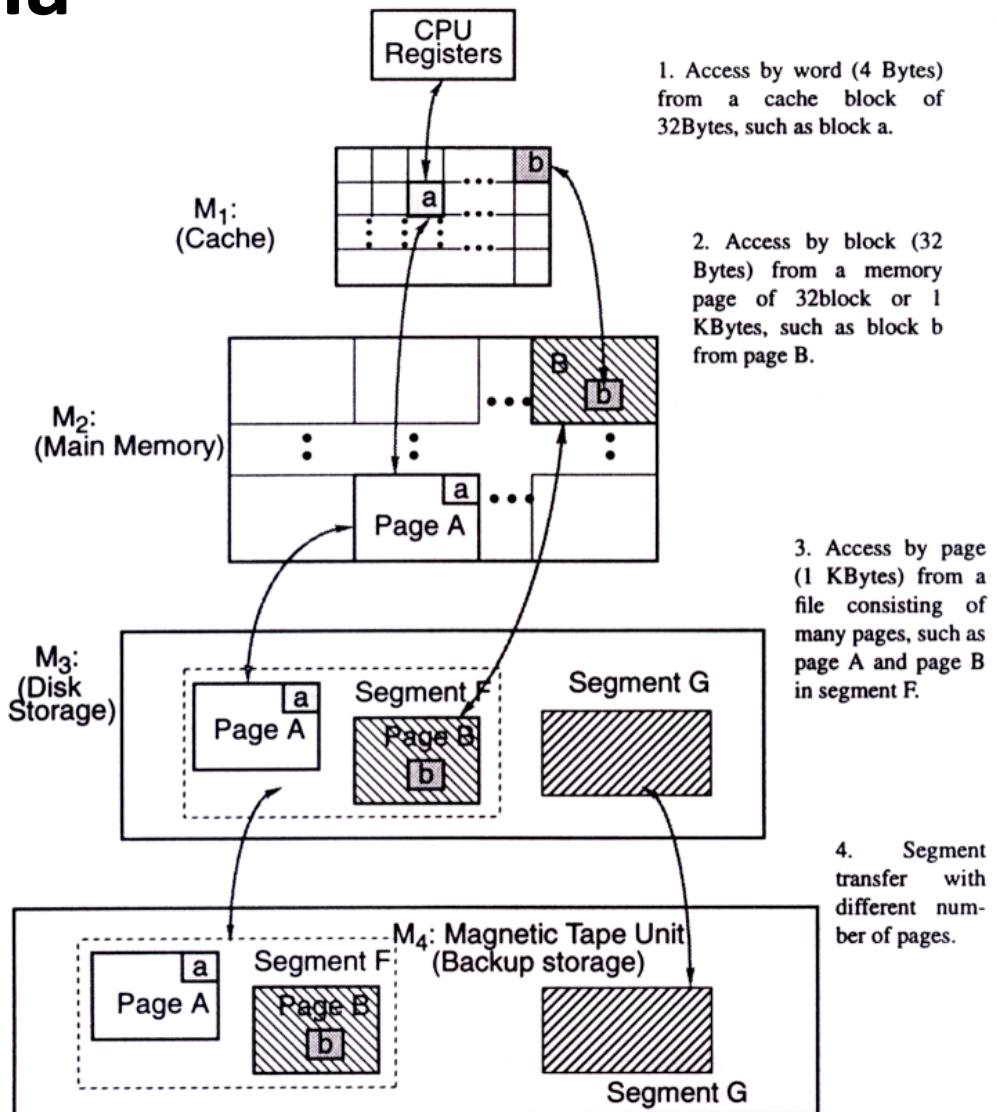
$$b_i > b_{i+1}$$

$$x_i < x_{i+1}$$

Jerarquía de memoria

■ Propiedad de inclusión

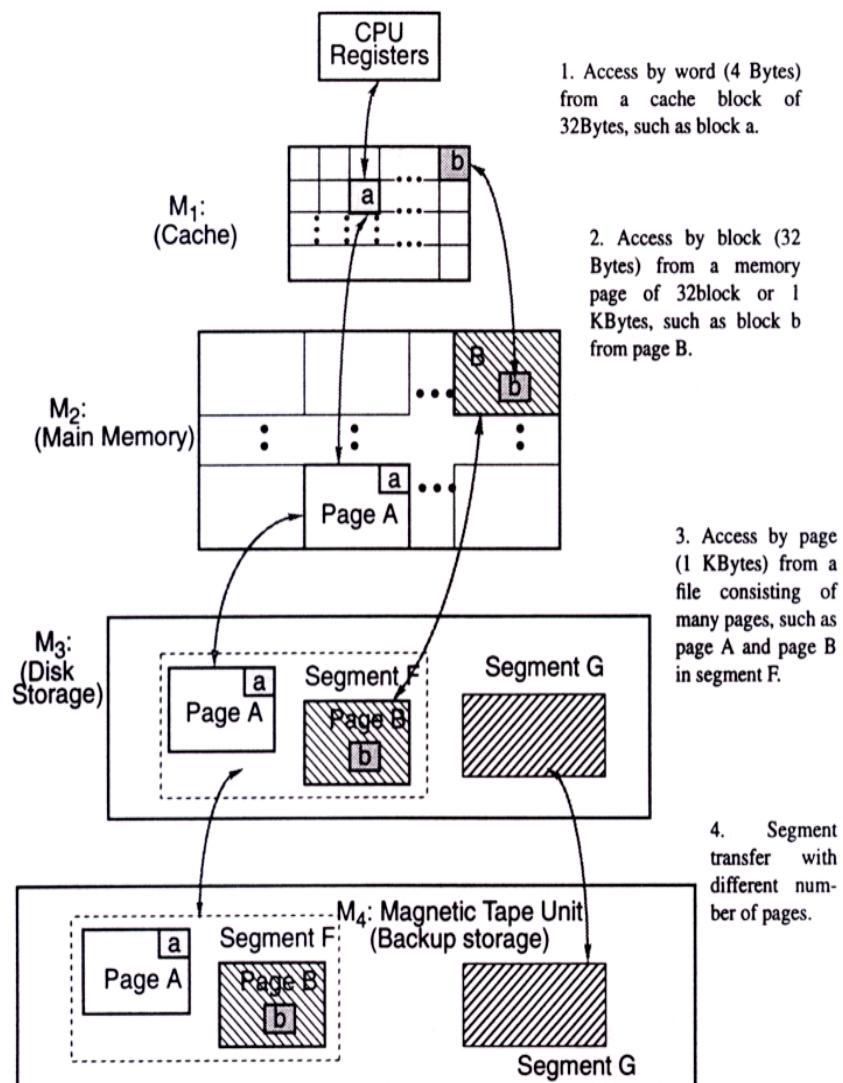
- $M_1 \subset M_2 \subset M_3 \subset \dots \subset M_n$
- Si una palabra se encuentra en $M_i \Rightarrow$ copias de esa palabra también se encuentran en $M_{i+1}, M_{i+2}, \dots, M_n$.
- Sin embargo, una palabra almacenada en M_{i+1} puede no estar en M_i , y si es así, tampoco estará en $M_{i-1}, M_{i-2}, \dots, M_1$.



Jerarquía de memoria

Lectura

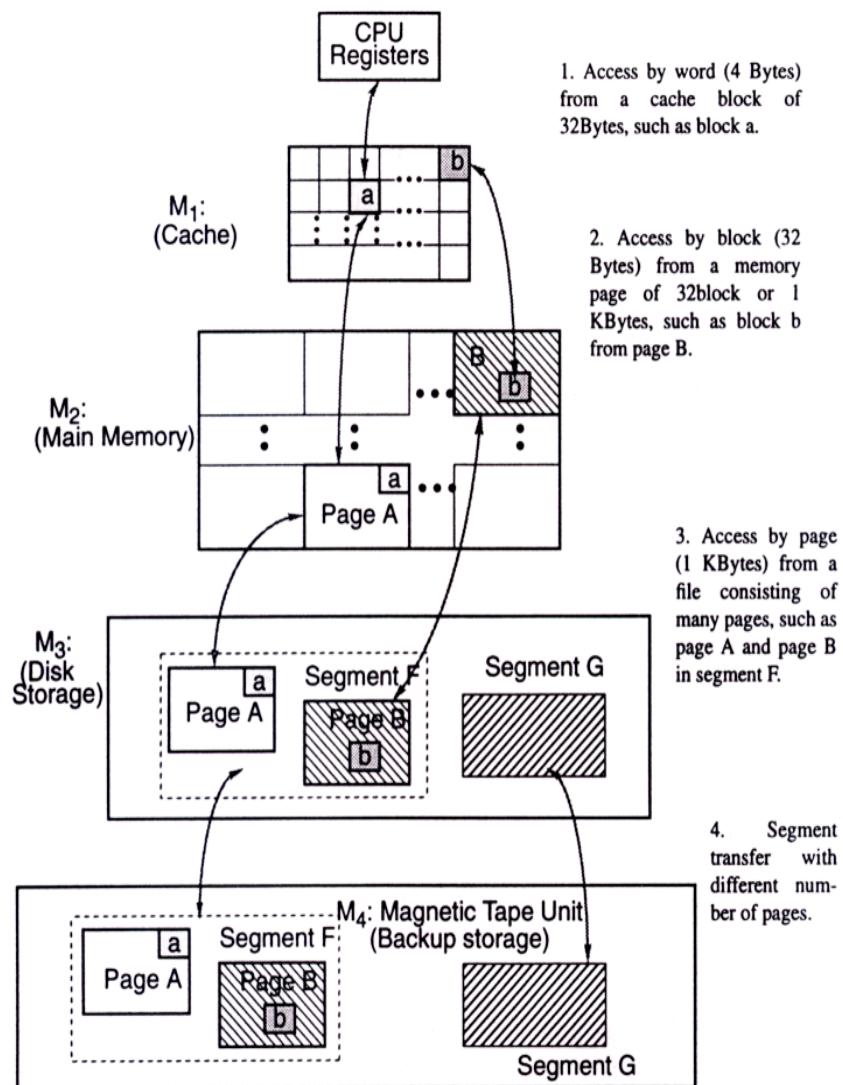
- No toda la información que necesita la CPU está en M_1 .
- Si la palabra deseada no se encuentra en el nivel 1 \Rightarrow se intentará localizarla en los niveles inferiores \Rightarrow Penalización en el tiempo de acceso.
- Supóngase que está en $M_j, j > 1$. Se transferirá a M_{j-1} , luego a M_{j-2} , y así sucesivamente hasta llegar al nivel 1, si bien pueden existir “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.



Jerarquía de memoria

■ 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 se usan dos estrategias para mantener la coherencia:
- **WRITE-THROUGH (Escritura directa):**
 - Si se modifica una palabra en $M_i \Rightarrow$ 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 de la jerarquía

■ Modelo de evaluación del rendimiento de una jerarquía de memoria (C. K. Chow, 1974*)

- Se suele considerar que la política de administración de memoria viene caracterizada por una función de éxito o tasa de aciertos A .
- Tasa de aciertos A_i (*hit ratio*)
 - Porcentaje de información buscada en un ciclo de memoria que está presente en el nivel i .
 - En una jerarquía con n niveles:
 - $A_0 = 0$
 - $A_n = 1$
 - A_i depende de:
 - Capacidad del nivel i (s_i).
 - Granularidad de la transferencia de información.
 - Estrategia de administración de memoria.

Modelo de evaluación de la jerarquía

- Tasa de fallos F_i : (*miss ratio*)
 - $F_i = 1 - A_i, i=0, \dots, n.$
 - $F_0 = 1$
 - $F_n = 0$
- Frecuencia de accesos con éxito al nivel i 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 $i-1$.
$$\sum_{i=1}^n a_i = 1$$
 - Dado que la información de M_j está también en $M_k, k > j$:
 - $a_i = A_i - A_{i-1}, i = 1, \dots, n$
 - $a_1 = A_1$
 - $a_n = 1 - A_{n-1}$

Modelo de evaluación de la jerarquía

- 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 Rendimiento:

- Se puede medir por el tiempo medio de acceso de la jerarquía para cada referencia a memoria ().

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

$$T_i = \sum_{j=1}^i t_j$$

t_j = tiempo de acceso medio individual del nivel j

$$\bar{T} = \sum_{i=1}^n a_i T_i$$

Sumatoria de la frecuencia de acceso con éxito a M_i , multiplicada por el tiempo de acceso efectivo a M_i ,

Modelo de evaluación de la jerarquía

$$\bar{T} = \sum_{i=1}^n a_i T_i$$

- Sustituyendo a_i y T_i :

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

Modelo de evaluación de la jerarquía

■ ②Coste por bit:

- El coste por bit promedio $c(n)$ de un sistema de n niveles es:

$$c(n) = \frac{\sum_{i=1}^n c_i s_i}{\sum_{i=1}^n s_i} + c_0$$

The equation is annotated with three yellow boxes:

- A box around $\sum_{i=1}^n c_i s_i$ labeled **coste total**.
- A box around $+ c_0$ labeled **coste de interconexión entre niveles**.
- A box around $\sum_{i=1}^n s_i$ labeled **tamaño total**.

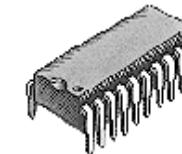
Problemas a resolver

- **Problemas a resolver en cualquier nivel de la jerarquía de memoria:**
 - **Colocación** o ubicación de bloques
 - ¿Dónde se ubicará un bloque en el nivel i cuando se transfiere a éste desde el $i+1$?
 - **Identificación** de bloques
 - ¿Cómo se encuentra un bloque en el nivel i ?
 - **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 se reemplaza?
 - **Estrategia de escritura**
 - ¿Qué acciones hay que llevar a cabo si una escritura modifica un bloque en el nivel superior?

Memoria

- **Jerarquía de memoria. Concepto de localidad**
- **Memorias RAM semiconductoras. Memorias de sólo lectura. Prestaciones: velocidad, tamaño y coste**
- **Configuración y diseño de memorias utilizando varios chips**
- **Memorias asociativas**
- **Memoria cache. Influencia en las prestaciones**

Memorias de semiconductor



- Todas son de acceso aleatorio
- Tipos de memoria:
 - De sólo lectura o **ROM** (*Read Only Memory*):
 - ROM, PROM, EPROM, EEPROM.
 - De lectura y escritura o **RAM** (*Random Access Memory*):
 - SRAM, DRAM.

ROM

■ De sólo lectura o ROM:

■ **ROM** (*Read Only Memory*)

- La información se graba en el proceso de fabricación mediante máscaras ⇒ no puede alterarse ni borrarse.
- Razonable económicamente sólo para muchos chips.
- Ej.: *firmware* en una calculadora, BIOS de VGA, ROM de autoarranque, ROM para microcontroladores, etc.

■ **PROM** (*Programmable ROM*)

- La información se graba en un proceso posterior irreversible ⇒ programable una sola vez.
 - Los chips se fabrican con todas las celdas conectadas (a 1) y la programación (puesta a 0) consiste en romper contactos haciendo pasar una corriente elevada.
- ✖ Tecnología bipolar ⇒ alto consumo y poca densidad de integración ⇒ obsoleta desde finales de los 80.

ROM



■ **EPROM (Erasable Programmable ROM)**

- Se pueden borrar exponiéndolas a radiación ultravioleta ⇒ programable varias veces (100 a 1000).
 - Tienen una ventana de cristal de cuarzo para permitir el borrado, que se suele tapar con un adhesivo una vez programada para evitar el borrado accidental debido a la exposición prolongada al sol.
- Los datos permanecen de 10 a 100 años.
- Ejs.: Memoria de programa para un microcontrolador, BIOS.
- **OTPROM (One Time Programmable ROM)**
 - EPROM sin ventana
 - ✓ más barata debido al empaquetado
 - ✗ programable sólo la primera vez

ROM

■ EEPROM (*Electrically Erasable Programmable ROM*)

- Se programan **byte a byte** mediante corrientes elevadas.



Programador de EEPROM

- ✓ Ventaja sobre EPROM: reprogramable en la misma placa.
- Retención de datos: 10 a 100 años.
- No puede considerarse memoria RAM:
 - Reprogramable muchas, pero limitadas, veces
 - » 10 000 a >1 000 000
 - Borrado y programación lentos (5-10 ms)
- Ejs.: almacenamiento de números en teléfono o fax, almacenamiento de configuración en un monitor.

ROM

■ Memoria FLASH

- EEPROM que se puede reprogramar **por bloques** (toda la memoria o un bloque se borra a la vez).
 - Mayor densidad.
 - Menor precio por bit.
- Retención de datos: >20 años.
- Ejs.: Discos de estado sólido USB (*pen-drives*), tarjetas de memoria, BIOS.



SRAM

■ De lectura y escritura o RAM:

- Estáticas o **SRAM** (*Static Random Access Memory*).
 - **Async. SRAM, Sync. SRAM, Pipeline Burst SRAM.**
 - Para caché
 - » por velocidad.
 - Para memoria principal de microcontroladores
 - » por requerir poca memoria,
 - » por no merecer la pena añadir la circuitería extra que requiere una DRAM.

...

SRAM

...

- **CAM** (*Content Addressable Memory*).
 - Memorias asociativas. *Las estudiaremos más adelante.*
- **NOVRAM** (*Non-Volatile RAM*):
 - SRAM con una pila de litio incorporada (retención: 5 a 10 años).
 - O bien híbrido entre SRAM y EEPROM.
 - » Cada celda SRAM tiene una celda EEPROM asociada.
 - » Al desconectarse la alimentación se copia el contenido de la SRAM en la EEPROM, y viceversa al conectarse.

...

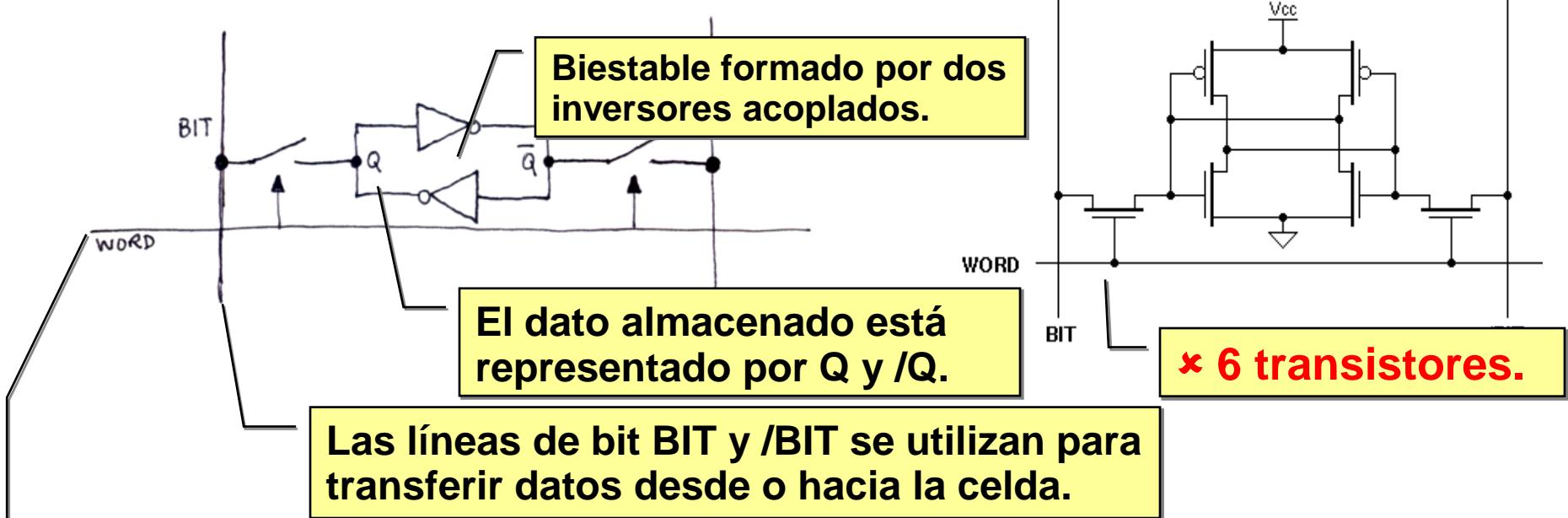
DRAM

- **Dinámicas**, o con refresco, o **DRAM** (*Dynamic RAM*).
 - Para memoria principal:
 - **PSRAM** (*Pseudo Static RAM*)
 - **FPM DRAM** (*Fast Page Mode*)
 - **Static Column DRAM**
 - **Nibble mode DRAM**
 - **DRAM EDO** (*Extended Data Out*)
 - **Burst DRAM** o **DRAM BEDO** (*Burst Extended Data Out*)
 - **EDRAM** (*Enhanced DRAM*)
 - **SDRAM** (*Synchronous DRAM*)
 - **SDRAM II** o **DDR SDRAM** (*Double Data Rate*)
 - **ESDRAM** (*Enhanced Synchronous DRAM*)
 - **RDRAM** (*Rambus DRAM*) o **DRDRAM** (*Direct Rambus DRAM*)
 - **DDR-II** y **DDR-III**
 - Para memoria de vídeo (tarjetas gráficas):
 - **VRAM** (*Video RAM*).
 - **WRAM** (*Window RAM*).
 - **SGRAM** (*Synchronous Graphic RAM*).

SRAM

■ Celda de memoria SRAM

- Estática ⇒ los datos almacenados se mantienen por un tiempo indefinido si hay alimentación.



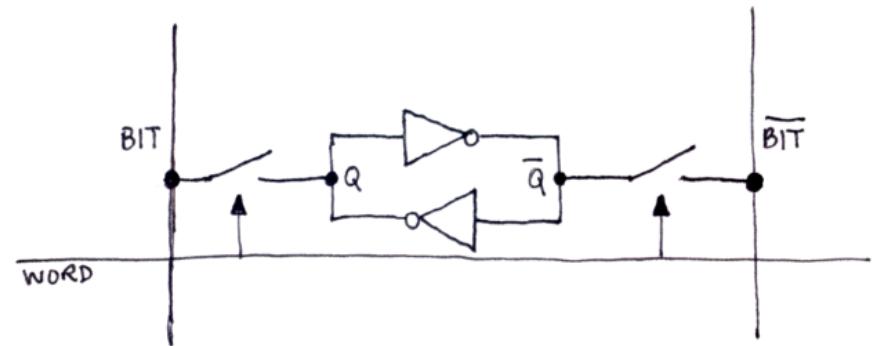
- Cuando la línea WORD esté a 1 se selecciona la celda para lectura o escritura.
- Cuando la línea WORD esté a 0 los dos transistores asociados no conducen, desconectando la celda de las líneas de bit.

SRAM

■ Celda de memoria SRAM

■ Operación de lectura:

- BIT y /BIT se ponen a 1 “débil”.
- Se selecciona la celda poniendo WORD a 1 \Rightarrow Q se conecta a BIT y /Q a /BIT.
- Si Q = 1 ($/Q = 0$) \Rightarrow la línea /BIT se pone a 0.
 - BIT = 1 y /BIT = 0 \Rightarrow se lee un 1.
- Si Q = 0 ($/Q = 1$) \Rightarrow la línea BIT se pone a 0.
 - BIT = 0 y /BIT = 1 \Rightarrow se lee un 0.



✓ Lectura no destructiva.

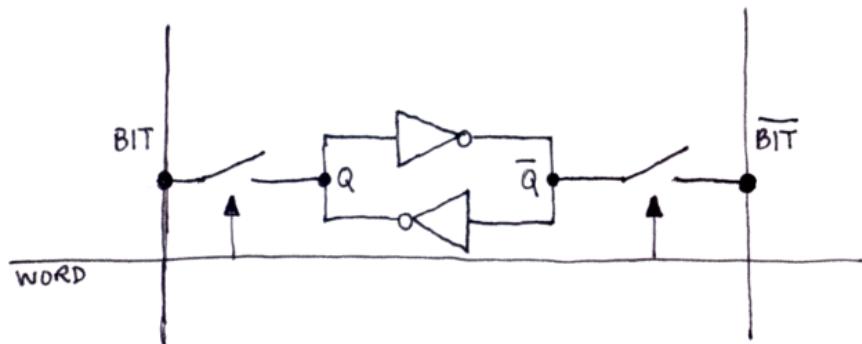
✓ Mayor velocidad que DRAM.

SRAM

■ Celda de memoria SRAM

■ Operación de escritura:

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

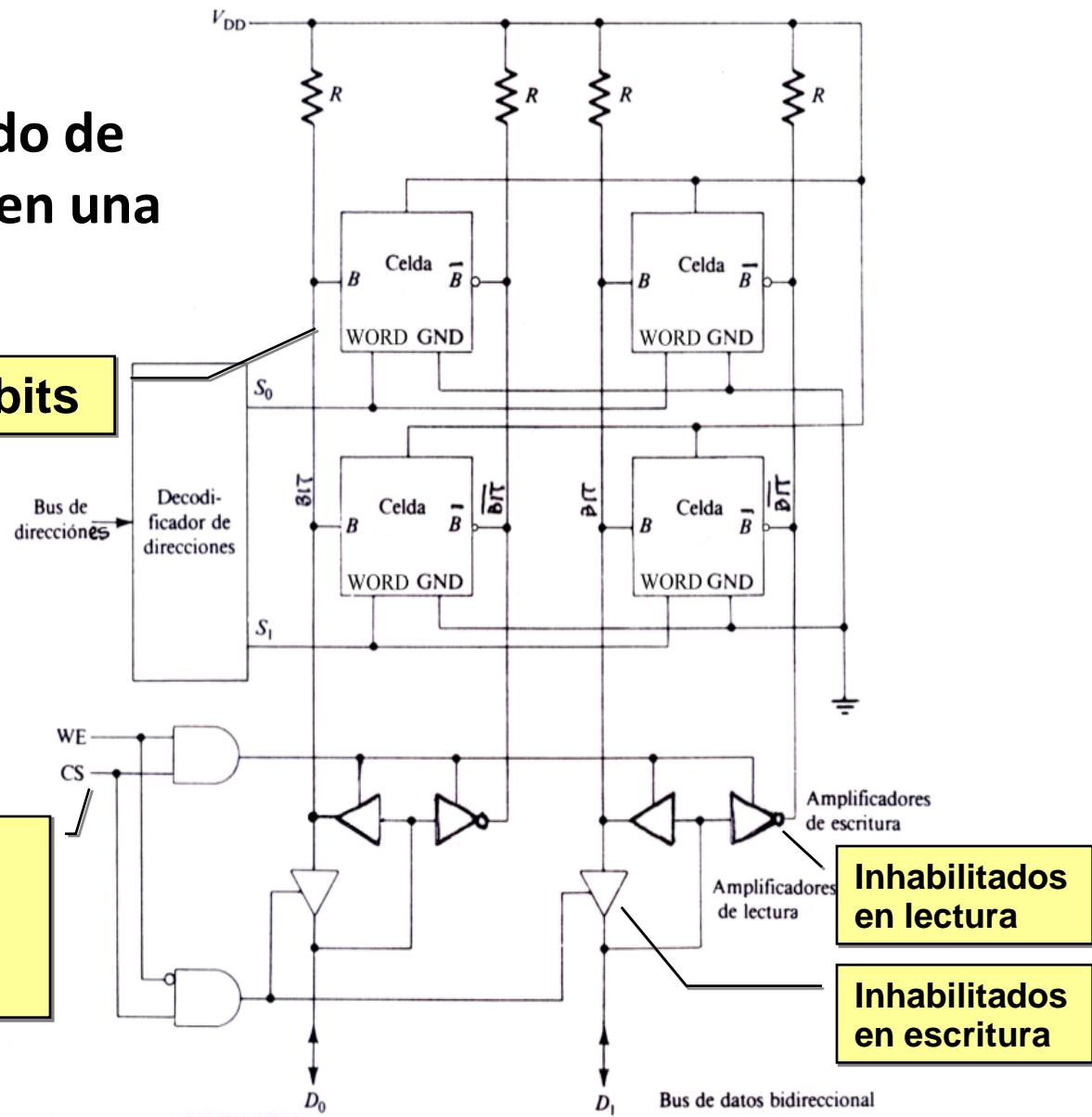


SRAM

- Esquema simplificado de conexión de celdas en una SRAM:

SRAM de 2 palabras de 2 bits

Si CS = 1 \Rightarrow
WE = 1 \Rightarrow Escritura
WE = 0 \Rightarrow Lectura



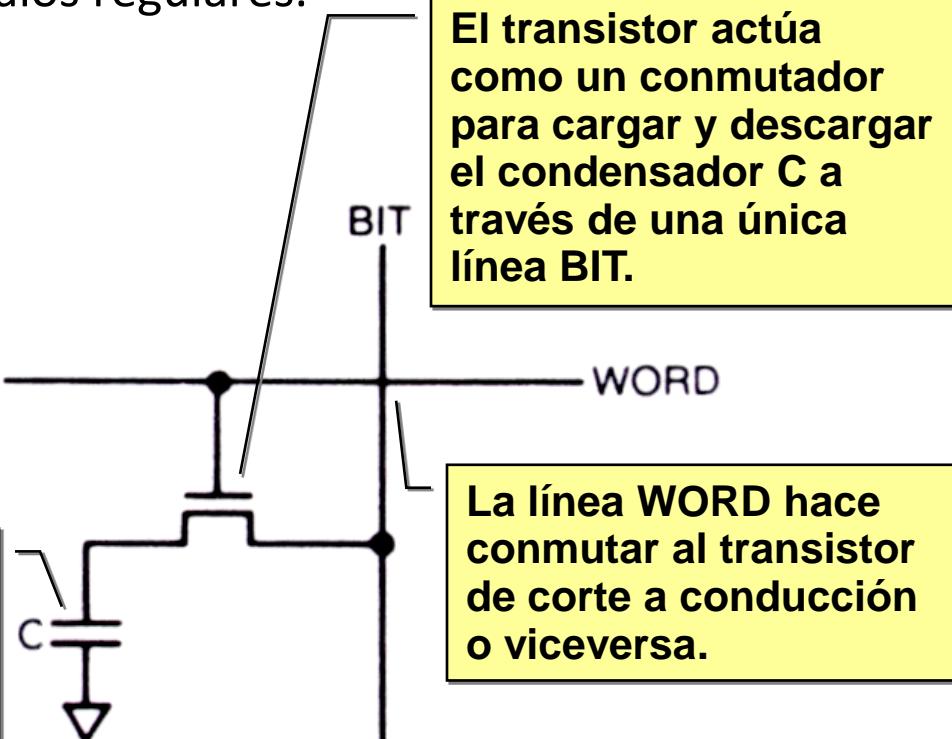
DRAM

■ Celda de memoria DRAM

- **Dinámica** ⇒ los datos almacenados decaen o se desvanecen y deben ser restaurados a intervalos regulares.

✓ Sencillez de la celda de almacenamiento.

La información se almacena en forma de carga eléctrica en un condensador C. La presencia (o ausencia) de carga indica que hay almacenado un 1 (o un 0).

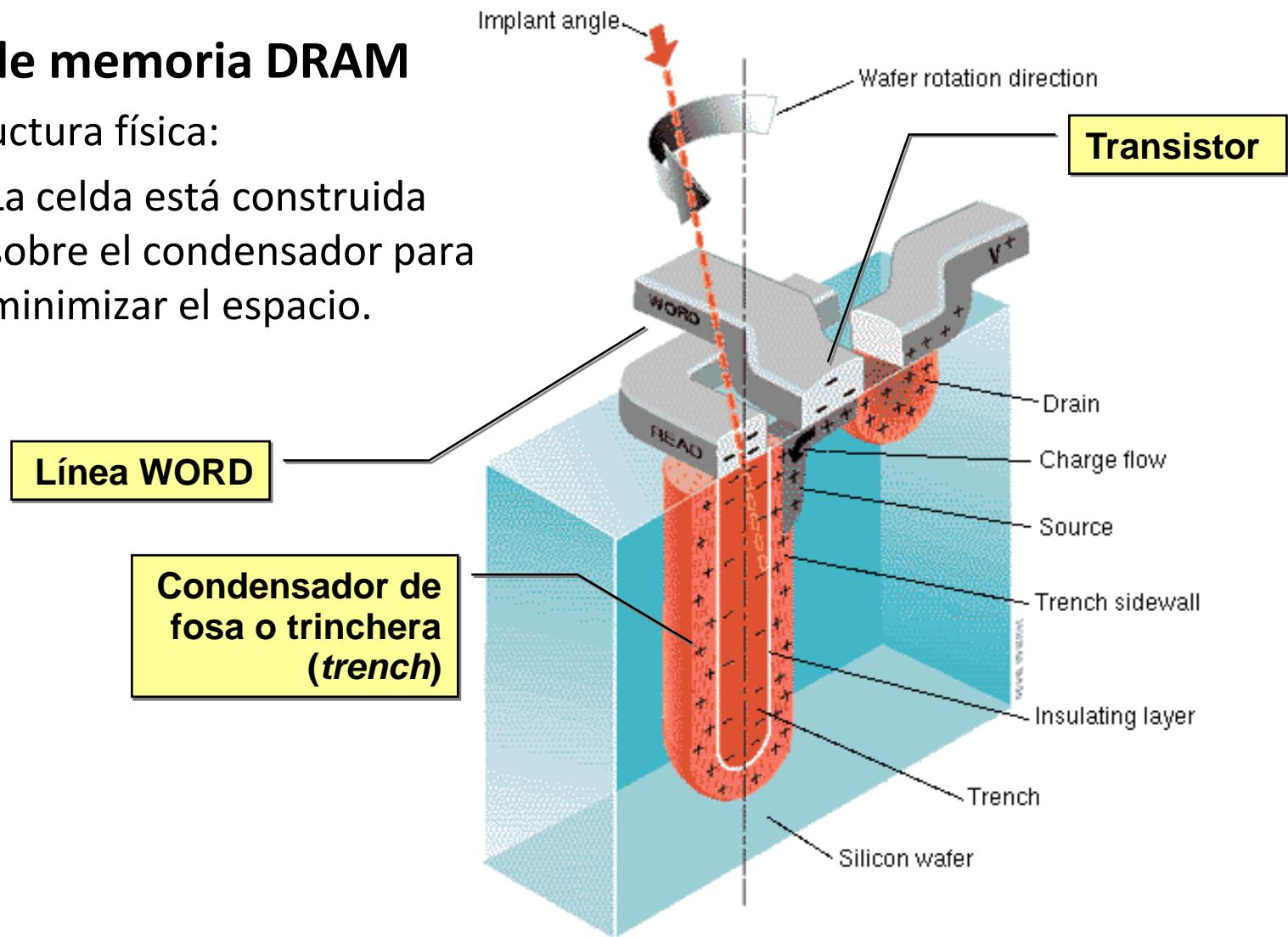


DRAM

■ Celda de memoria DRAM

■ Estructura física:

- La celda está construida sobre el condensador para minimizar el espacio.

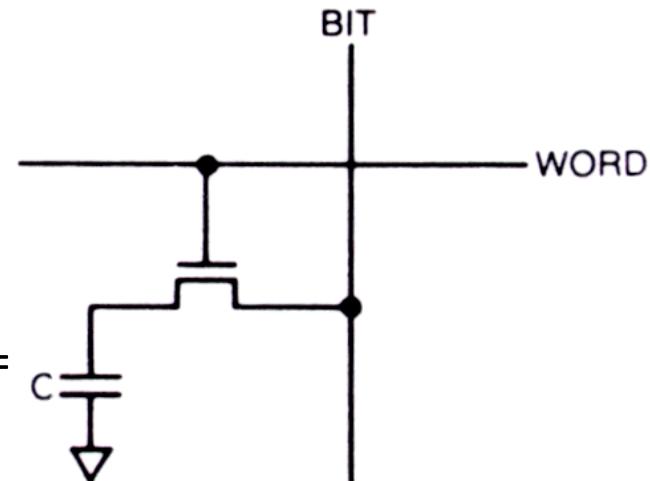


DRAM

■ Celda de memoria DRAM

▪ Operación de lectura:

- La circuitería externa convierte a BIT en una línea de salida, seleccionándose la celda con WORD = 1.
- Si C está cargado (= 1) \Rightarrow se descarga a través de la línea BIT \Rightarrow se produce un pulso de corriente que es detectado por un amplificador de salida ("*sense amplifier*") \Rightarrow aparece un 1 en la línea de datos de salida.
- Si C está descargado (= 0) \Rightarrow no se produce pulso de corriente \Rightarrow aparece un 0 en la línea de datos de salida.



✗ Lectura destructiva.

↓
Debe ir seguida de una escritura que restaure el estado original.

↓
Realizada automáticamente por los circuitos de control de la DRAM

DRAM

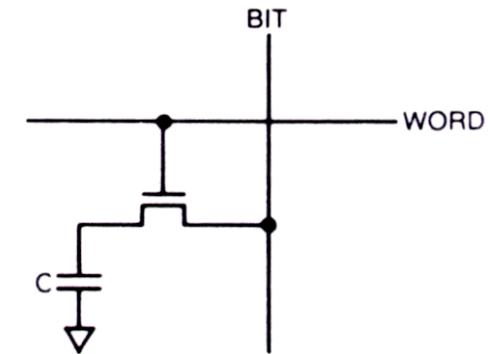
■ Celda de memoria DRAM

■ Operación de escritura:

- Se usa BIT como entrada y se pone a 0 ó a 1, seleccionándose la celda que tenga WORD a 1.
- Si BIT = 1 \Rightarrow C se carga.
- Si BIT = 0 \Rightarrow C se descarga.

■ Operación de refresco:

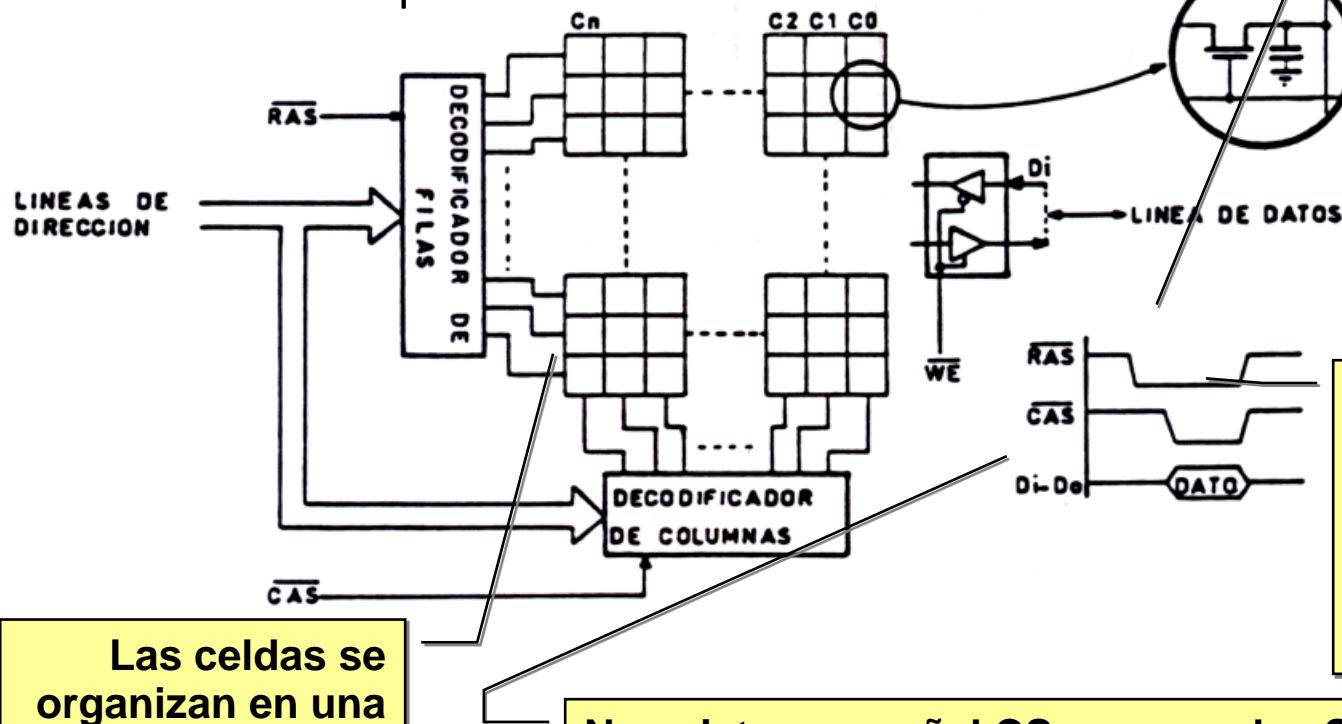
- El aislamiento del condensador C no es perfecto \Rightarrow su carga tiende a desvanecerse en pocos ms.
- El contenido de la DRAM ha de ser restaurado periódicamente. Puede haber un circuito de control externo que genere secuencialmente todas las direcciones, especificando una operación de lectura para cada dirección.



DRAM

Circuitos de memoria DRAM

- Capacidad elevada de los chips de memoria DRAM ⇒ las direcciones han de proporcionarse multiplexadas en el tiempo.



Las celdas se organizan en una matriz lo más cuadrada posible.

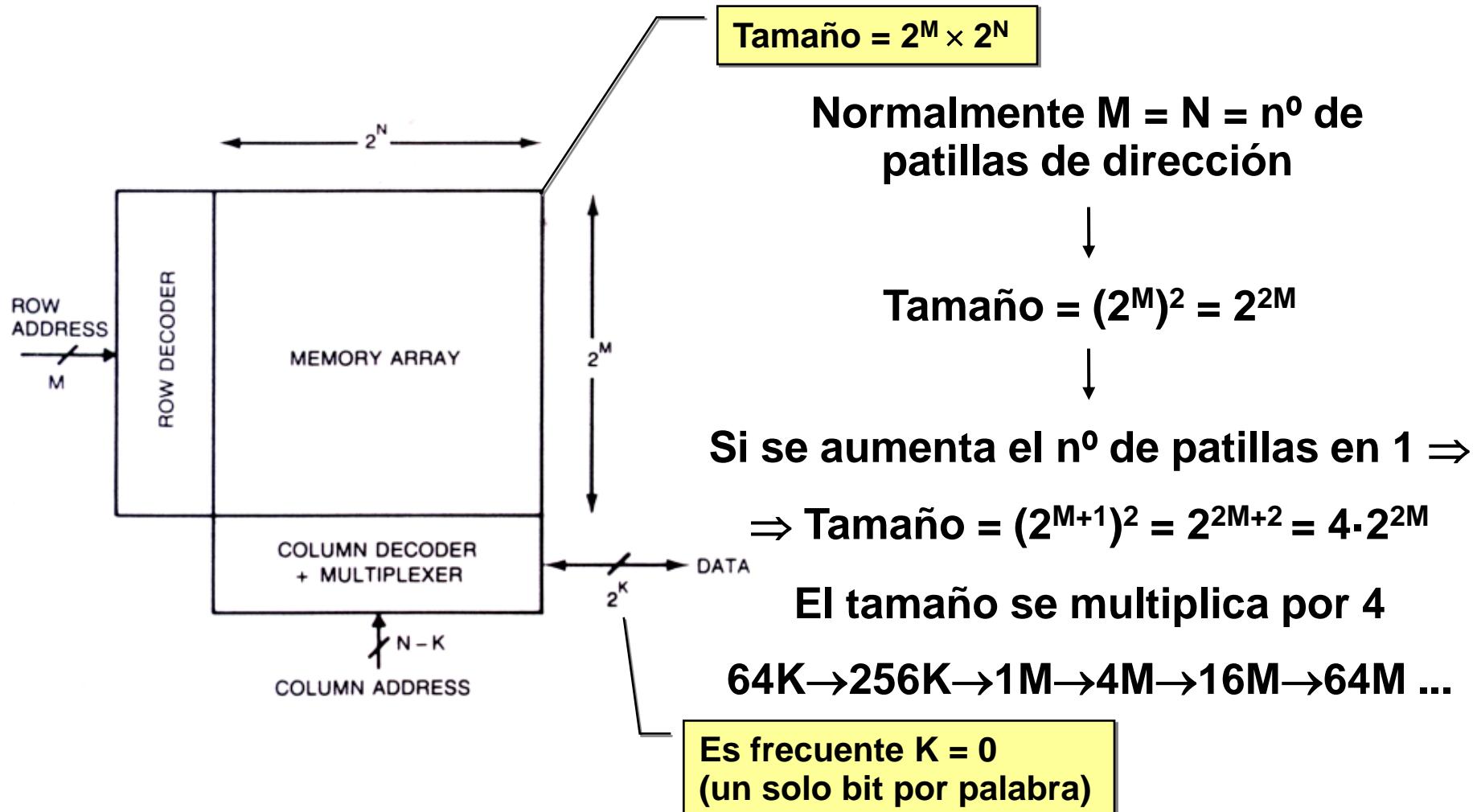
No existe una señal CS como en las SRAM. Para dar por válido un acceso a memoria es necesaria la secuencia completa /RAS - /CAS junto con la señal /WE (Write Enable).

Los bits correspondientes a las filas de la matriz de memoria se proporcionan en primer lugar, validándose por la señal /RAS (Row Access Strobe).

Después se proporcionan los bits que direccionan las columnas, validados por /CAS (Column Access Strobe).

DRAM

■ Circuitos de memoria DRAM



DRAM

- Tiempo de acceso.
- Tiempo de ciclo.

$t_{RAC} = t_a$
(tiempo de acceso):

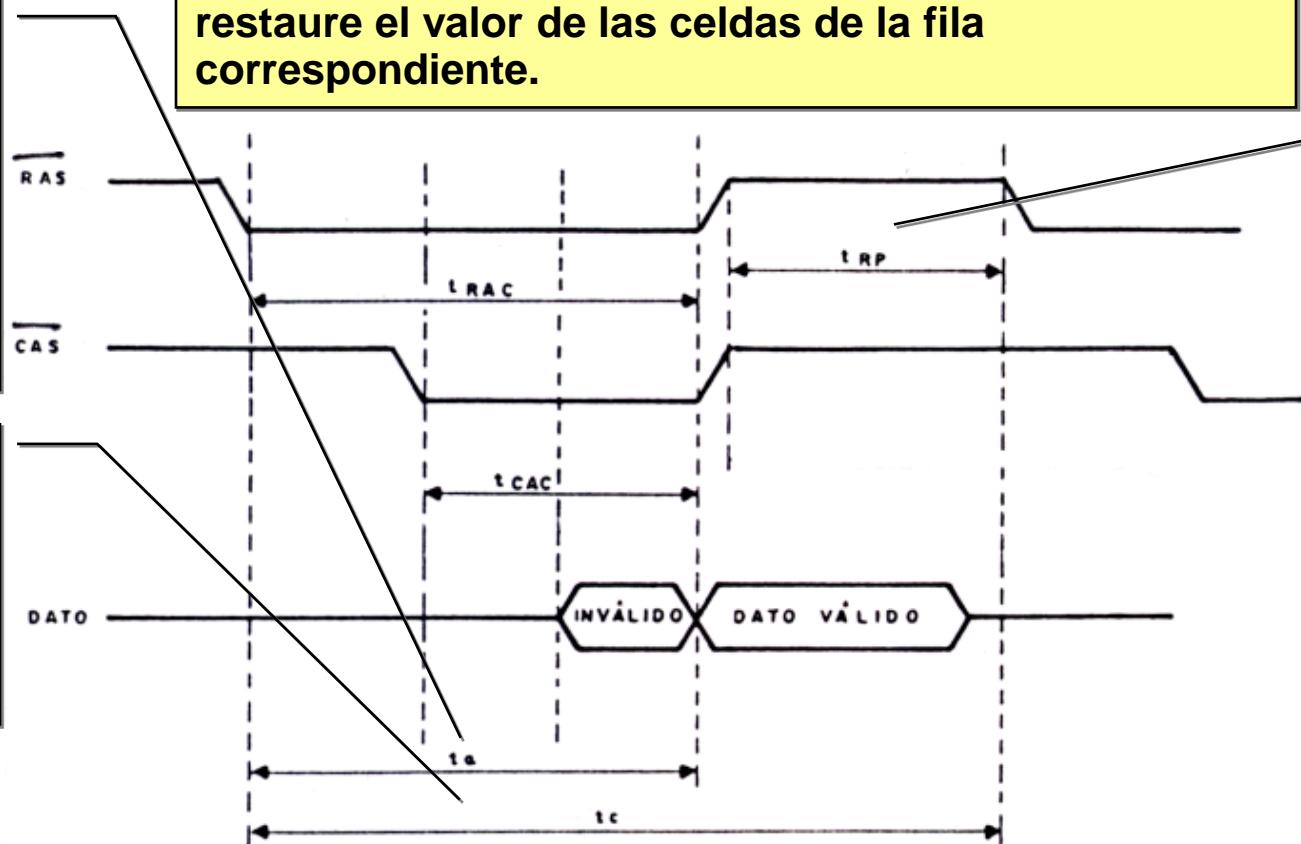
tiempo desde que se inicia una lectura ($/RAS \downarrow$) hasta que el dato está disponible en el bus de datos.

t_c (tiempo de ciclo):
 tiempo mínimo entre dos accesos consecutivos.

$$t_c \approx t_{RAC} + t_{RP}$$

$$t_c > t_a$$

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



DRAM

- Ejemplos de tiempos de acceso y de ciclo:

Año de introducción	Tamaño del chip	Acceso a filas		Acceso a columna (CAS)	Tiempo de ciclo
		DRAM más lenta	DRAM más rápida		
1980	64 Kbit	180 ns	150 ns	75 ns	250 ns
1983	256 Kbit	150 ns	120 ns	50 ns	220 ns
1986	1 Mbit	120 ns	100 ns	25 ns	190 ns
1989	4 Mbit	100 ns	80 ns	20 ns	165 ns
1992?	16 Mbit	≈ 85 ns	≈ 65 ns	≈ 15 ns	≈ 140 ns

Tiempos de las DRAM rápidas y lentas de cada generación.

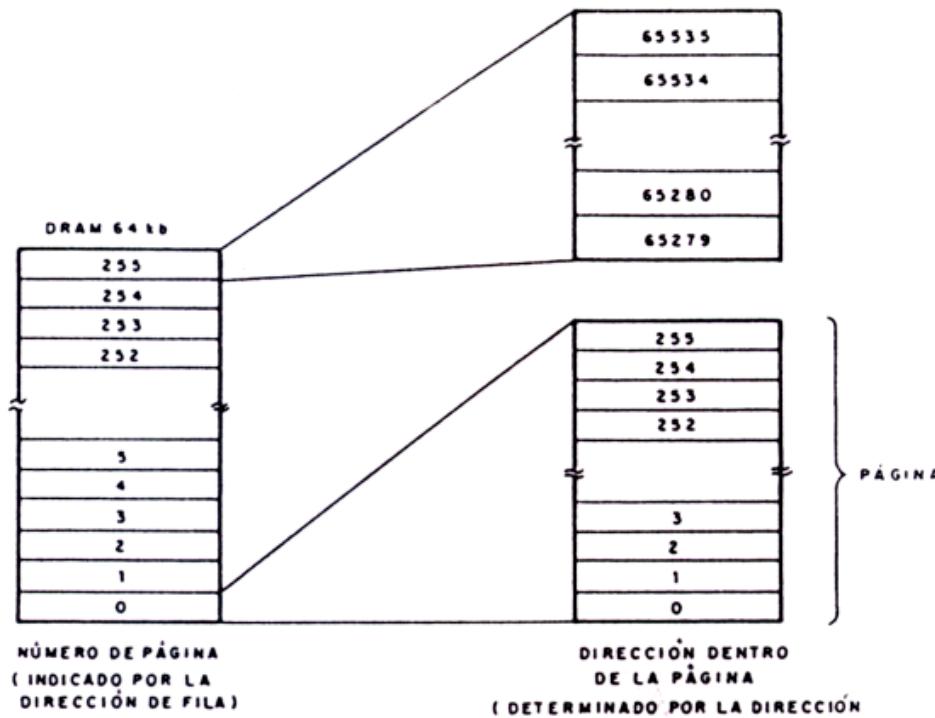
La mejora por un factor de dos en los accesos a columnas se produjo junto con el cambio de DRAM NMOS a DRAM CMOS. Con tres años por generación, la mejora de rendimiento del tiempo de acceso a filas es aproximadamente el 7 por 100 por año. Los datos de la última fila representan el rendimiento predicho para las DRAM de 16 Mbits, que no están todavía disponibles.

Hoy ~~no~~ están disponibles.

DRAM

■ DRAM con modo página rápida (*Fast Page Mode RAM, FPM RAM*):

- Se puede acceder a cualquier dirección (columna) dentro de una página (fila), una vez seleccionada esta última, manteniendo /RAS a 0 y cambiando /CAS de 1 a 0 en cada acceso.
- Estrictamente hablando, no es una memoria RAM ya que los accesos a varias columnas de la última fila son más rápidos.



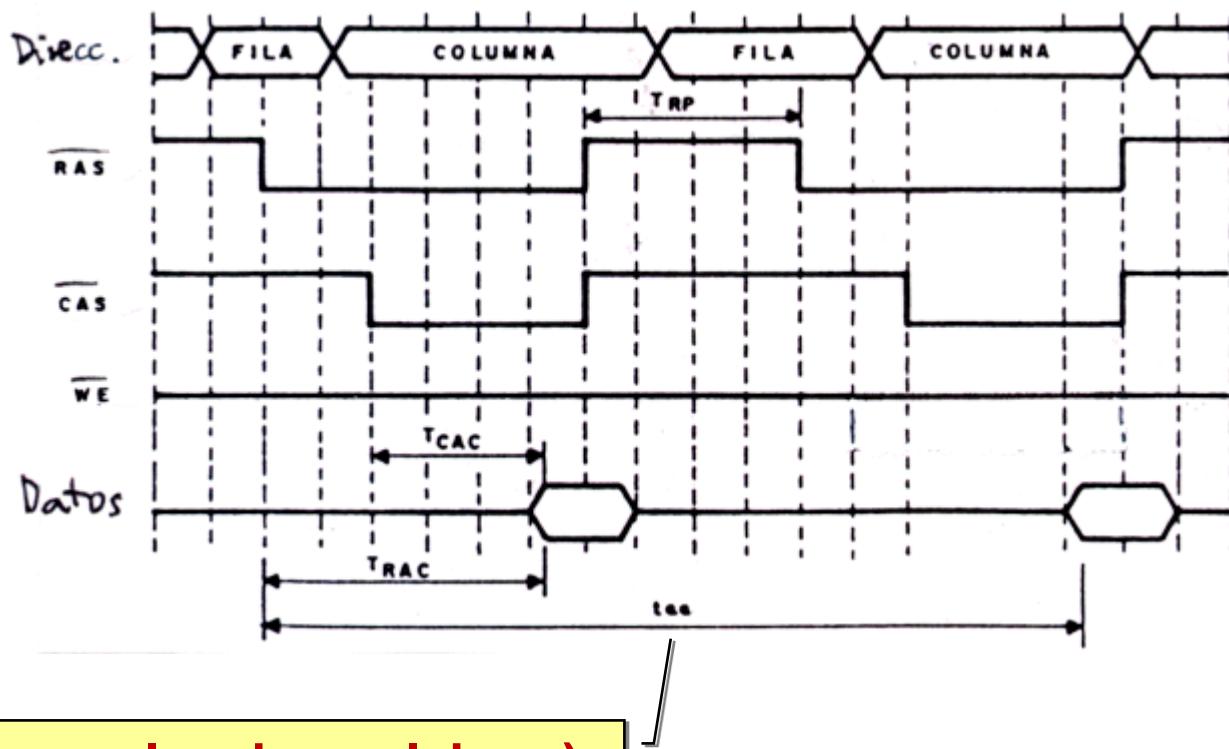
✓ **Velocidad ↑ (en comparación con acceso normal) .**

Hoy día (2011):

- ✗ **es la más lenta de todas las DRAM existentes.**
- ✗ **es cara debido a su baja demanda.**

DRAM

- Acceso a dos palabras:



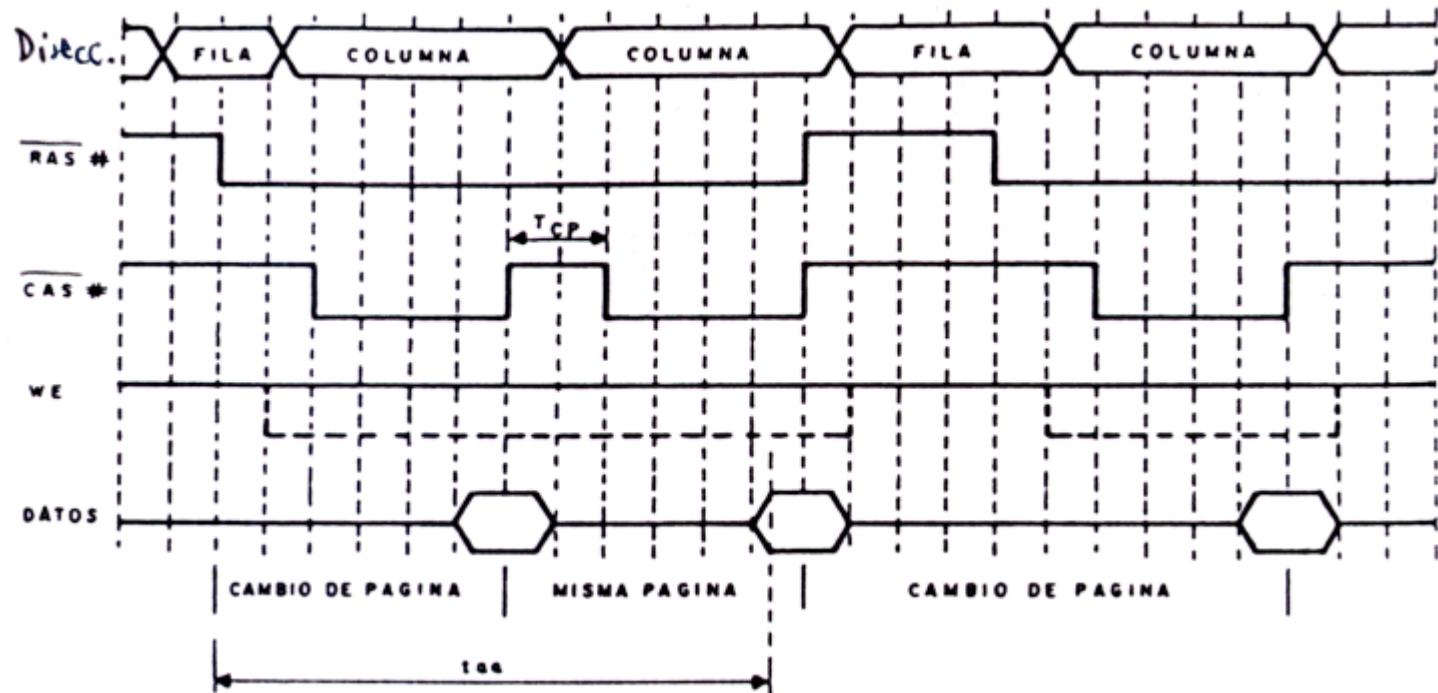
t_{aa} (tiempo de acceso a las dos palabras):

$$t_{aa} \approx t_{RAC} + t_{RP} + t_{RAC}$$

Ej.: $t_{aa} \approx 80 \text{ ns} + 60 \text{ ns} + 80 \text{ ns} = 220 \text{ ns}$

DRAM

- Acceso en modo página rápido (*Fast Page Mode*):



t_{aa} (tiempo de acceso a dos palabras en modo página):

$$t_{aa} \approx t_{RAC} + t_{CP} + t_{CAC}$$

Ej.: $t_{aa} \approx 80 \text{ ns} + 10 \text{ ns} + 20 \text{ ns} = 110 \text{ ns}$

DRAM

■ Refresco de DRAM

- Los chips DRAM refrescan automáticamente la fila accedida en cualquier ciclo de lectura o escritura, pero...
- ...se precisa una circuitería auxiliar que produzca ciclos de refresco, consistentes en recargar los condensadores de todas las celdas.
- Los ciclos de refresco deben producirse cada pocos ms
 - de 4 a 64 ms aproximadamente.
- El ciclo de refresco consiste en dar un impulso /RAS junto con la dirección de fila para todas las filas. Con ello se refrescan todas las columnas de cada fila.

DRAM

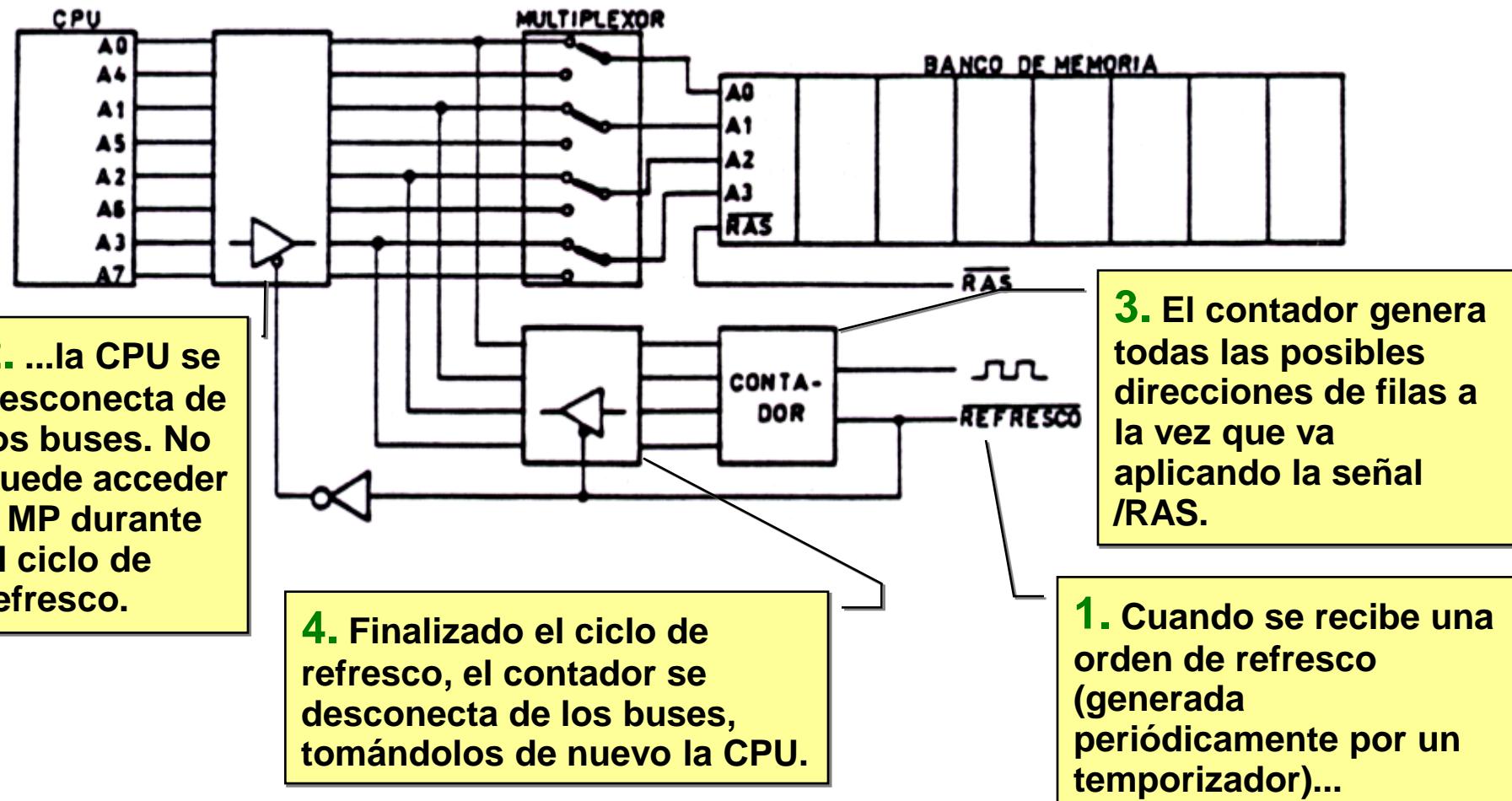
■ Refresco de DRAM

- La circuitería de refresco está basada fundamentalmente en un contador:
 - Controlador de DMA, o
 - Circuito controlador de memoria DRAM:
 - Genera las señales /RAS y /CAS.
 - Sirve de interfaz entre las señales que genera la CPU y el bus del sistema.
 - Controla el refresco.

DRAM

■ Refresco de DRAM

- Esquema genérico del circuito de refresco:



DRAM

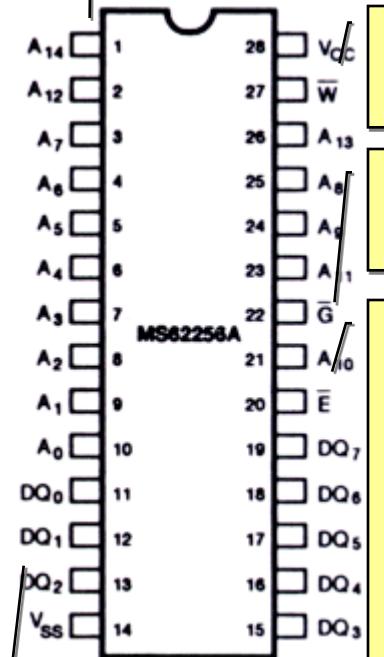
■ Refresco de DRAM

- Otra forma de hacer el refresco consiste en refrescar una única fila cada vez.
 - ✓ La CPU permanece menos tiempo seguido sin acceso al bus.
 - ✗ El refresco ha de pedirse con mayor periodicidad.
- Los chips DRAM actuales contienen:
 - Contador de refresco.
 - Modos de refresco que apenas necesitan circuitería externa.

Ejemplo de SRAM

256 K bits (32 K palabras × 8 bits)

A₀-A₁₄: 15 señales de dirección para seleccionar una de 32768 palabras de 8 bits.



/W (Write Enable) ≡ /WE. Se utiliza para indicar lectura o escritura.

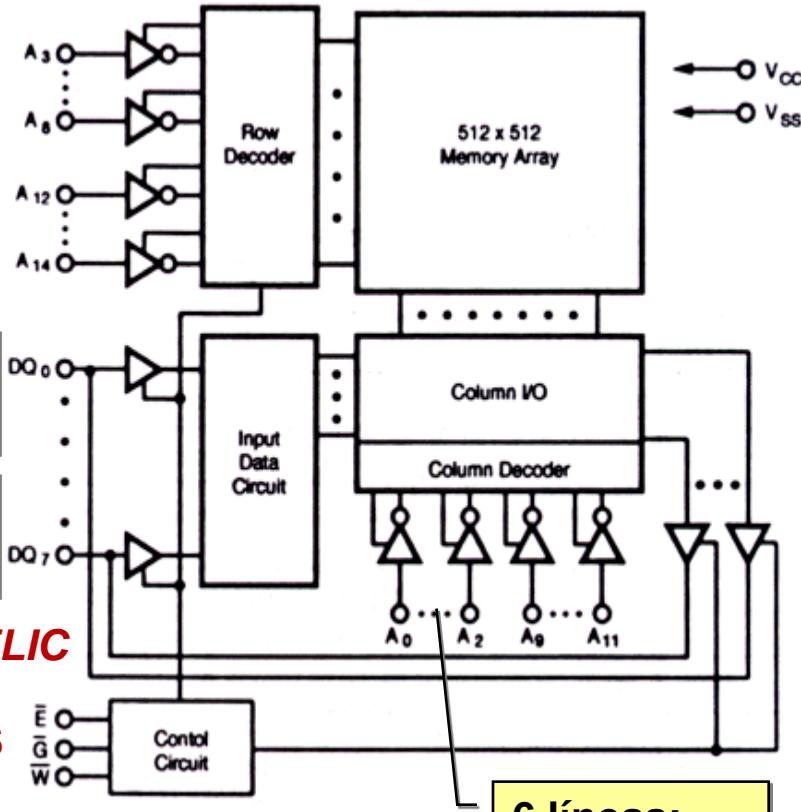
/G (Output Enable) ≡ /OE. Se selecciona (=0) para leer del chip.

/E (Chip Enable) ≡ /CS. Si no está activa ⇒ el chip no está seleccionado y permanece en “standby”, pasando su consumo de 900 mW a 50 mW.

DQ₀-DQ₇ (Data Input/Output)

**MOSEL-VITELIC
MS62256A
SRAM CMOS
32K × 8**

Mode	\bar{E}	\bar{G}	\bar{W}	I/O Operation
Standby	H	X	X	High Z
Read	L	L	H	D_{OUT}
Output Disabled	L	H	H	High Z
Write	L	X	L	D_{IN}



6 líneas:
 $9 - 6 = 3 \Rightarrow$
 se lee o
 escribe una
 palabra de
 $2^3 = 8$ bits

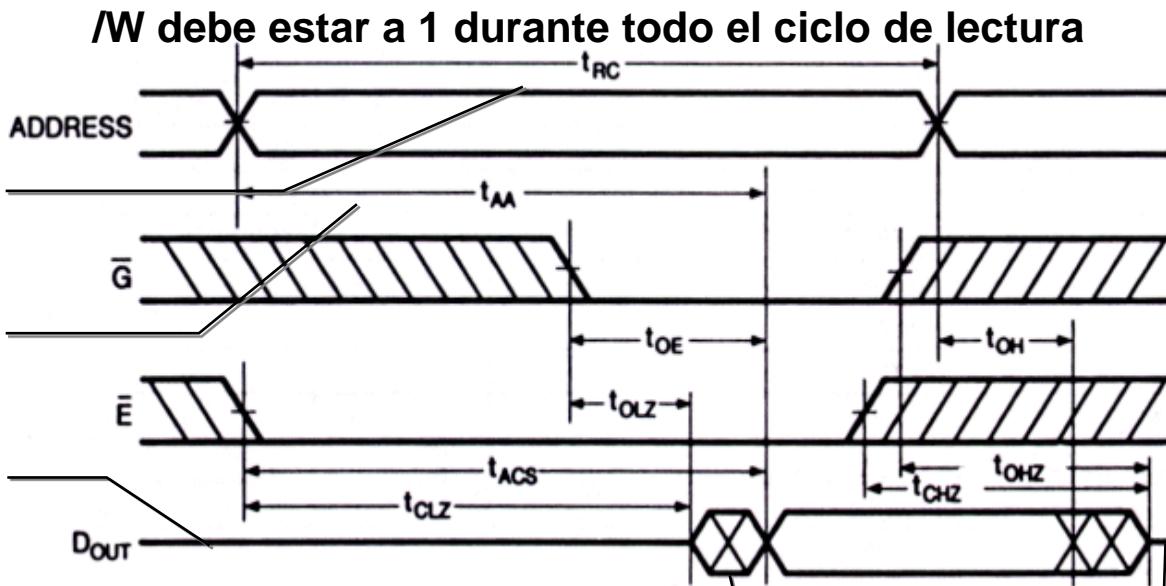
Ejemplo de SRAM

- Ciclo de lectura:**

t_{RC} : Tiempo de ciclo de lectura

t_{AA} : Tiempo de acceso

Alta impedancia



Parameter Name	Parameter	MS62256A-20		MS62256A-25		MS62256A-35		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
t_{RC}	Read Cycle Time	20	-	25	-	35	-	ns
t_{AA}	Address Access Time	-	20	-	25	-	35	ns
t_{ACS}	Chip Enable Access Time	-	20	-	25	-	35	ns
t_{OE}	Output Enable to Output Valid	-	8	-	12	-	15	ns
t_{OLZ}	Chip Enable to Output Low Z	3	-	5	-	5	-	ns
t_{OLZ}	Output Enable to Output in Low Z	0	-	0	-	0	-	ns
t_{CHZ}	Chip Disable to Output in High Z	-	8	-	10	-	15	ns
t_{OHZ}	Output Disable to Output in High Z	-	7	-	10	-	15	ns
t_{OHZ}	Output Hold from Address Change	3	-	5	-	5	-	ns

Cambiando.
Estado
desconocido.

Alta impedancia

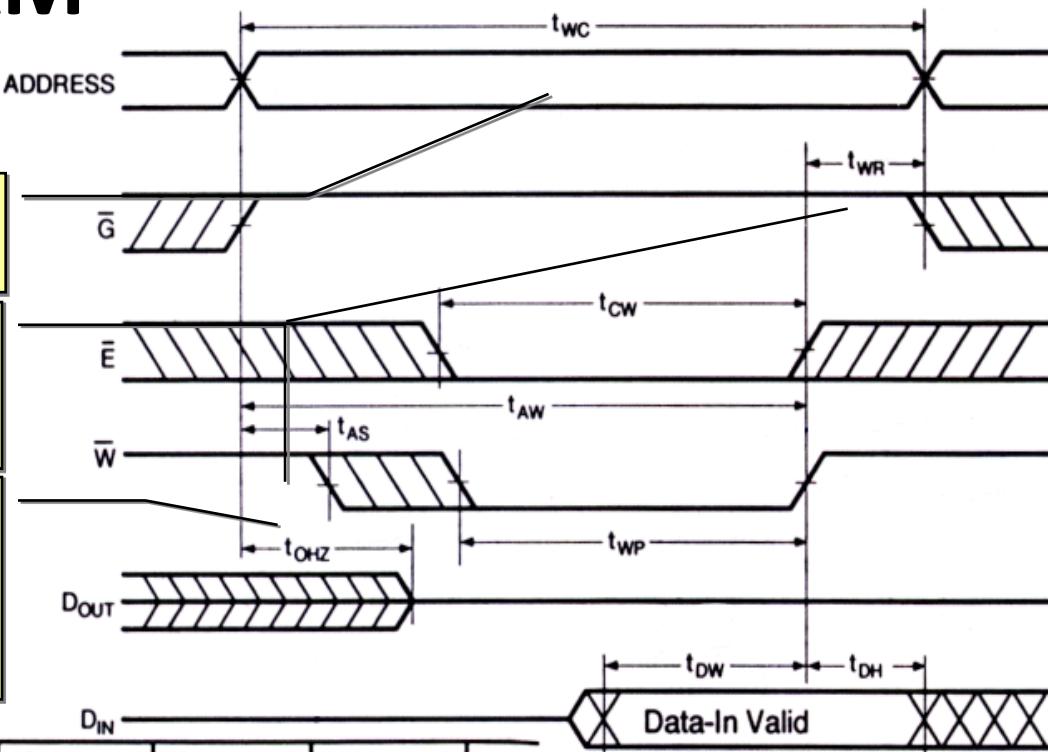
Ejemplo de SRAM

- Ciclo de escritura:**

t_{WC} : Tiempo de ciclo de escritura

Para garantizar que /W sea 1 durante la transición de dirección.

Durante t_{OHZ} las líneas D_{IN} no deben fijarse a 0 ni 1 (han de permanecer en alta impedancia).



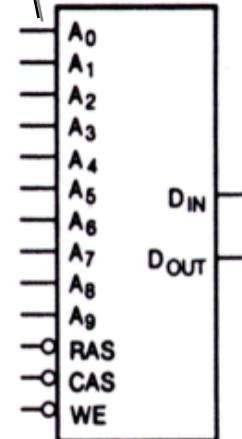
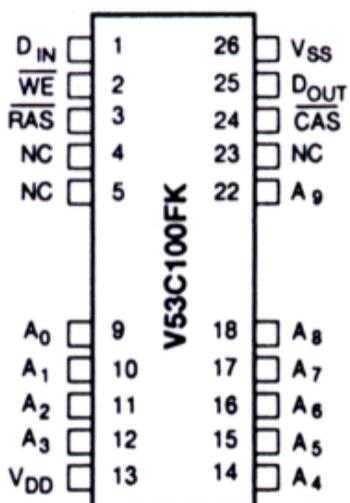
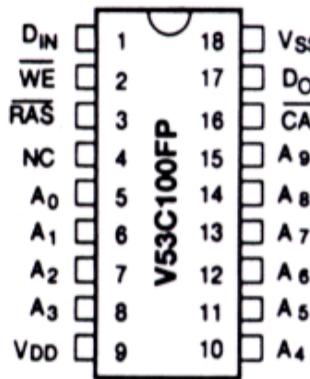
Parameter Name	Parameter	MS62256A-20		MS62256A-25		MS62256A-35		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
t_{WC}	Write Cycle Time	20	-	25	-	35	-	ns
t_{CW}	Chip Enable to End of Write	15	-	20	-	25	-	ns
t_{AS}	Address Set up Time	0	-	0	-	0	-	ns
t_{AW}	Address Valid to End of Write	15	-	20	-	25	-	ns
t_{WP}	Write Pulse Width	15	-	15	-	20	-	ns
t_{WR}	Write Recovery Time	0	-	0	-	0	-	ns
t_{OHZ}	Write to Output in High Z	0	8	0	13	0	15	ns
t_{DW}	Data to Write Time Overlap	10	-	13	-	15	-	ns
t_{DH}	Data Hold from Write Time	0	-	0	-	0	-	ns

Ejemplo de DRAM

FPM 1 M bit (1 M palabras × 1 bit)

Una dirección (20 bits) se divide (multiplexada en el tiempo) en 10 bits para la fila y 10 para la columna.

Tiempo de acceso: 60 ns
 Tiempo de ciclo: 120 ns
 T. de ciclo en modo página: 40 ns

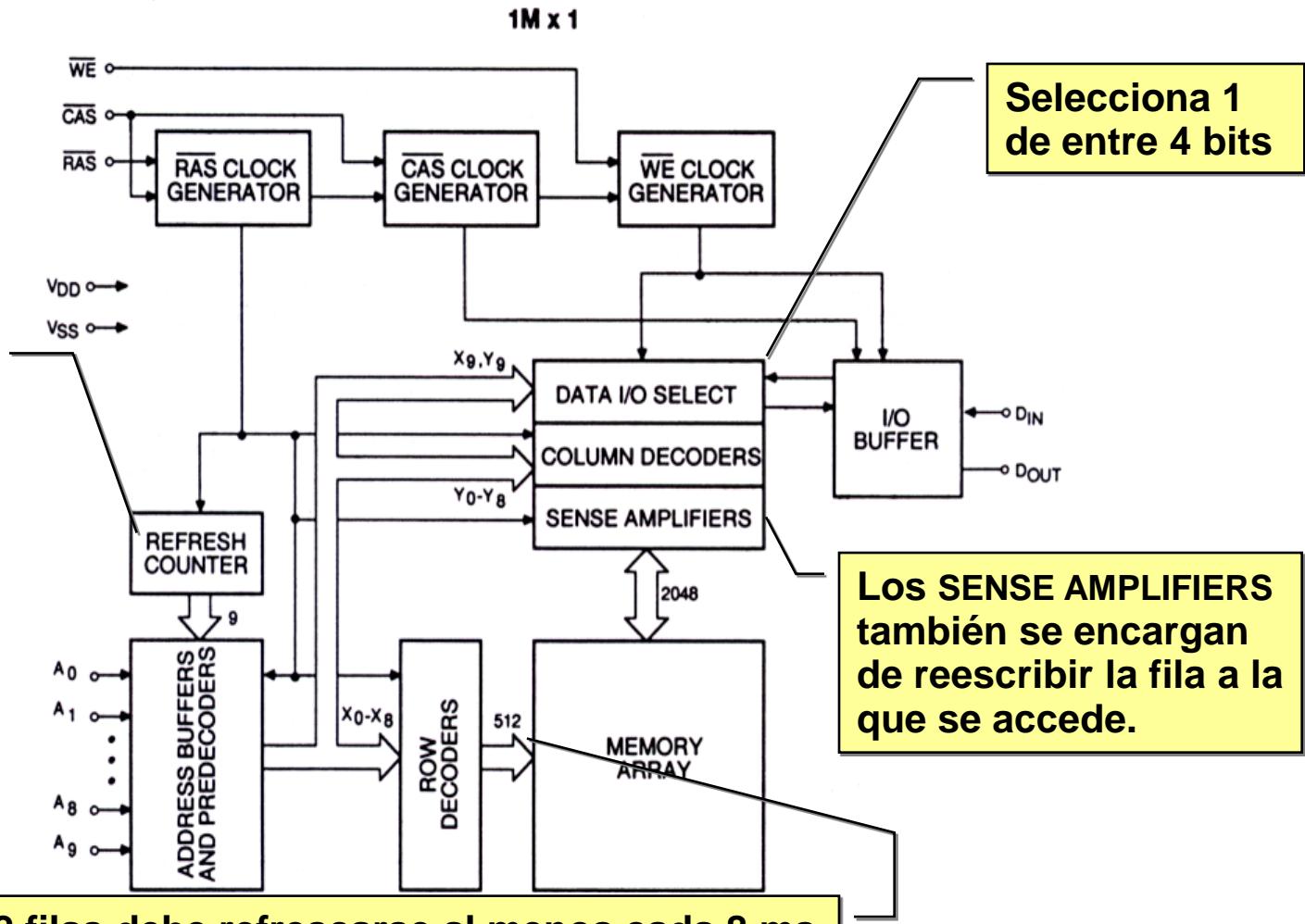


$A_0 - A_9$	Address Inputs
\overline{RAS}	Row Address Strobe
\overline{CAS}	Column Address Strobe
\overline{WE}	Write Enable
D_{IN}	Data Input
D_{OUT}	Data Output
V_{DD}	+5V Supply
V_{SS}	0V Supply
NC	No Connect

MOSEL-VITELIC
FAMILIA V53C100F
DRAM CMOS
FAST PAGE MODE
1M × 1 BIT

Ejemplo de DRAM

- Diagrama de bloques funcionales:

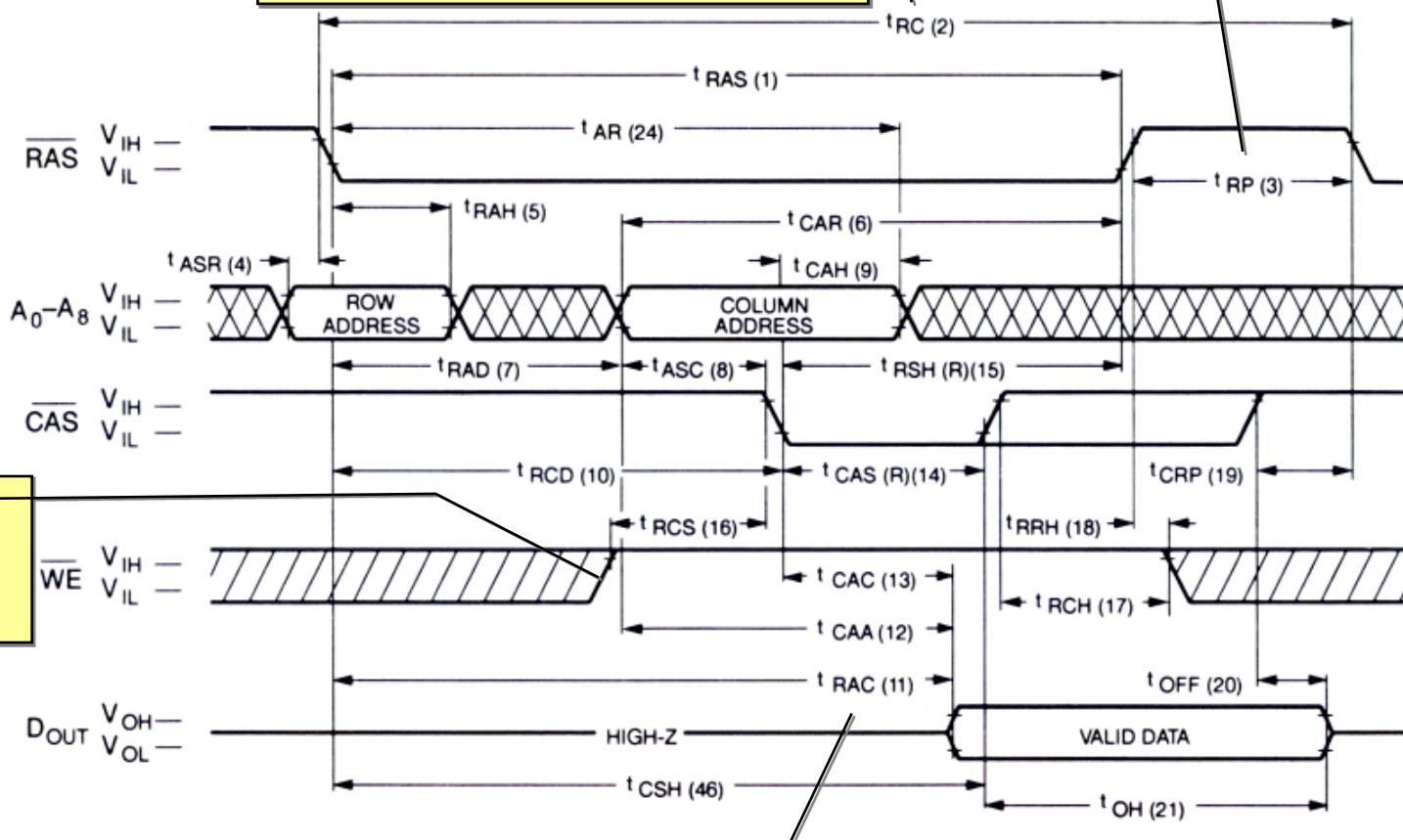


Ejemplo de DRAM

- Ciclo de lectura:**

t_{RP} : Tiempo de precarga de fila (empleado en refrescar la fila a accedida) (50 ns)

t_{RC} : tiempo de ciclo (120 ns)



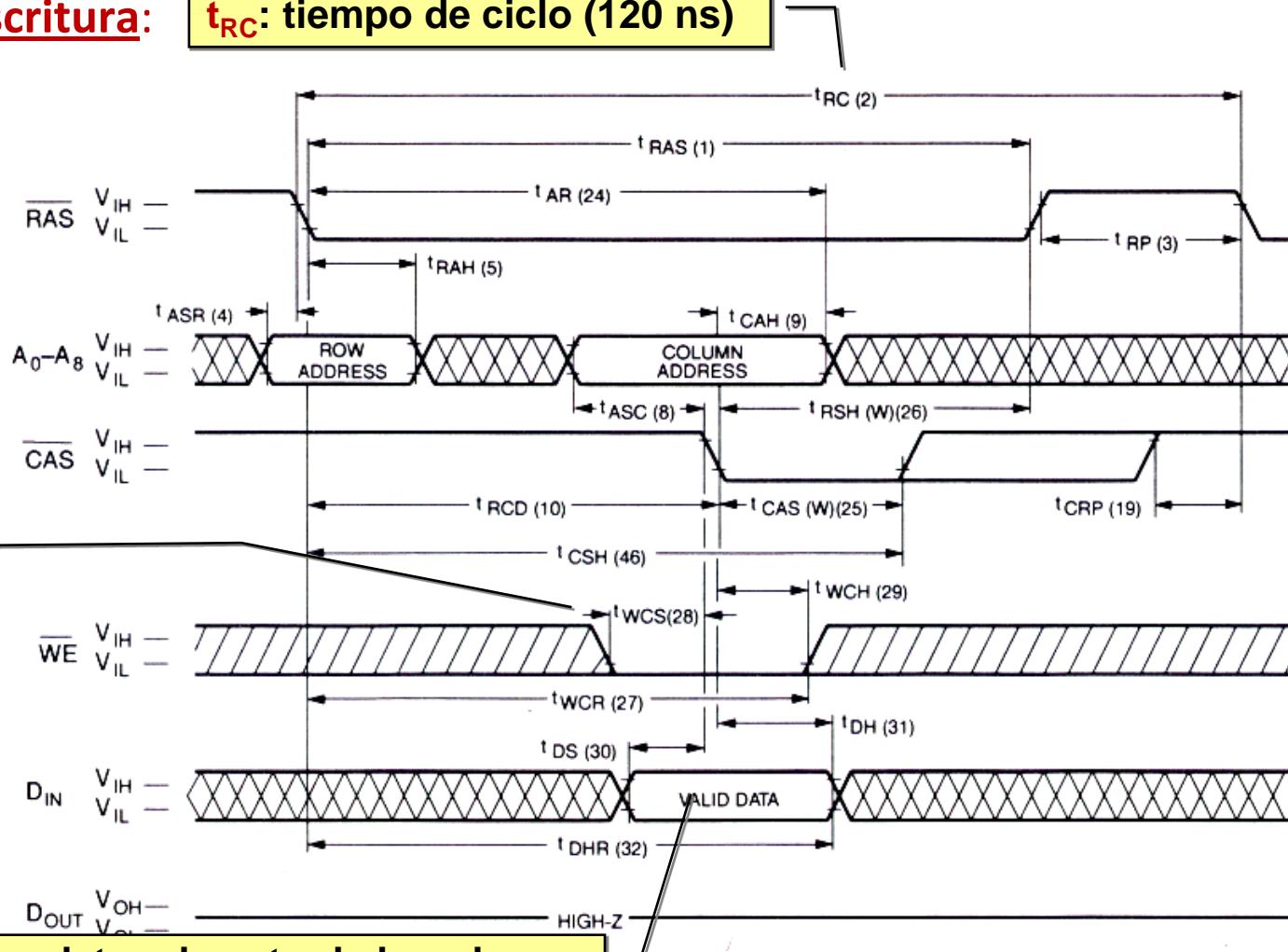
/WE ha de estar a 1 cuando /CAS↓

t_{RAC} : tiempo de acceso desde /RAS↓ (60 ns)

Ejemplo de DRAM

- Ciclo de escritura:**

t_{RC} : tiempo de ciclo (120 ns)



/WE ha de
estar a 0
cuando /CAS↓

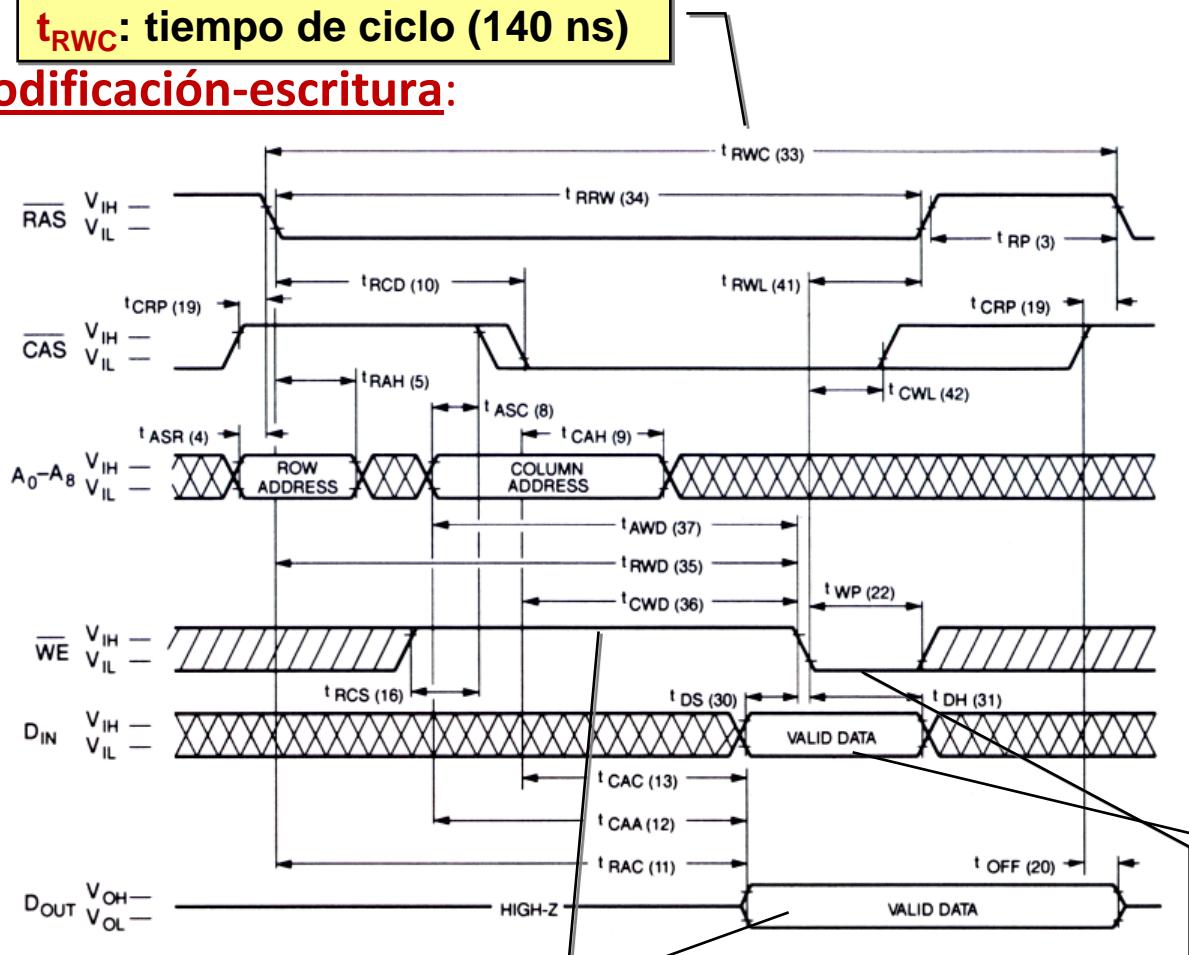
Los datos de entrada han de ser
válidos cuando /WE=0 y /CAS↓

Ejemplo de DRAM

t_{RWC} : tiempo de ciclo (140 ns)

- Ciclo de lectura-modificación-escritura:**

Permite el acceso para lectura a un bit y su posterior escritura, en la misma dirección, en un solo ciclo de duración ligeramente superior a la de un ciclo de lectura o escritura.



Primero se lee ($/WE=1$)...

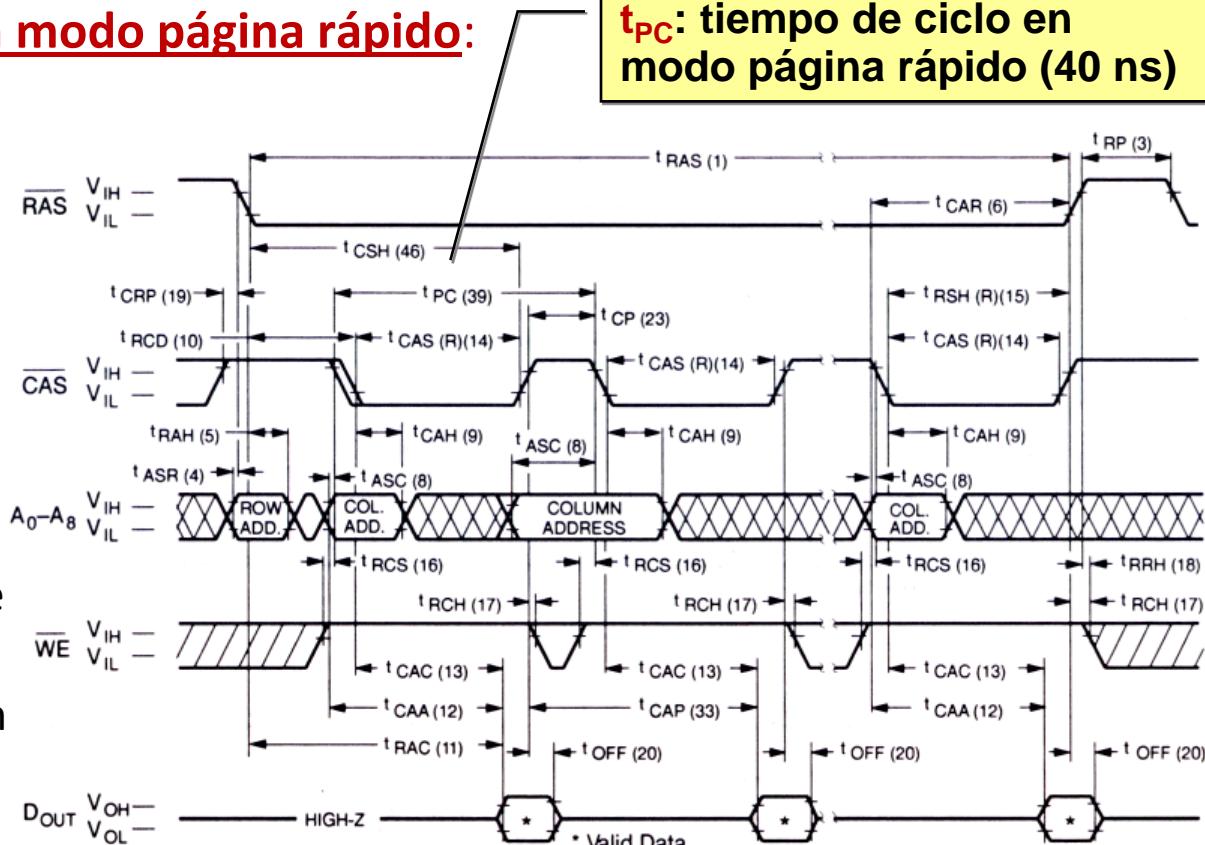
...y después se escribe ($/WE=0$).

Ejemplo de DRAM

- #### ▪ Ciclo de lectura en modo página rápida:

t_{PC}: tiempo de ciclo en modo página rápida (40 ns)

- Las 1024 columnas de una fila (página) pueden accederse a una frecuencia más alta, manteniendo /RAS a 0 mientras se suceden ciclos /CAS.
 - También existen ciclos de escritura y lectura-modificación-escritura en modo página rápido.



La máxima frecuencia de acceso a datos será:

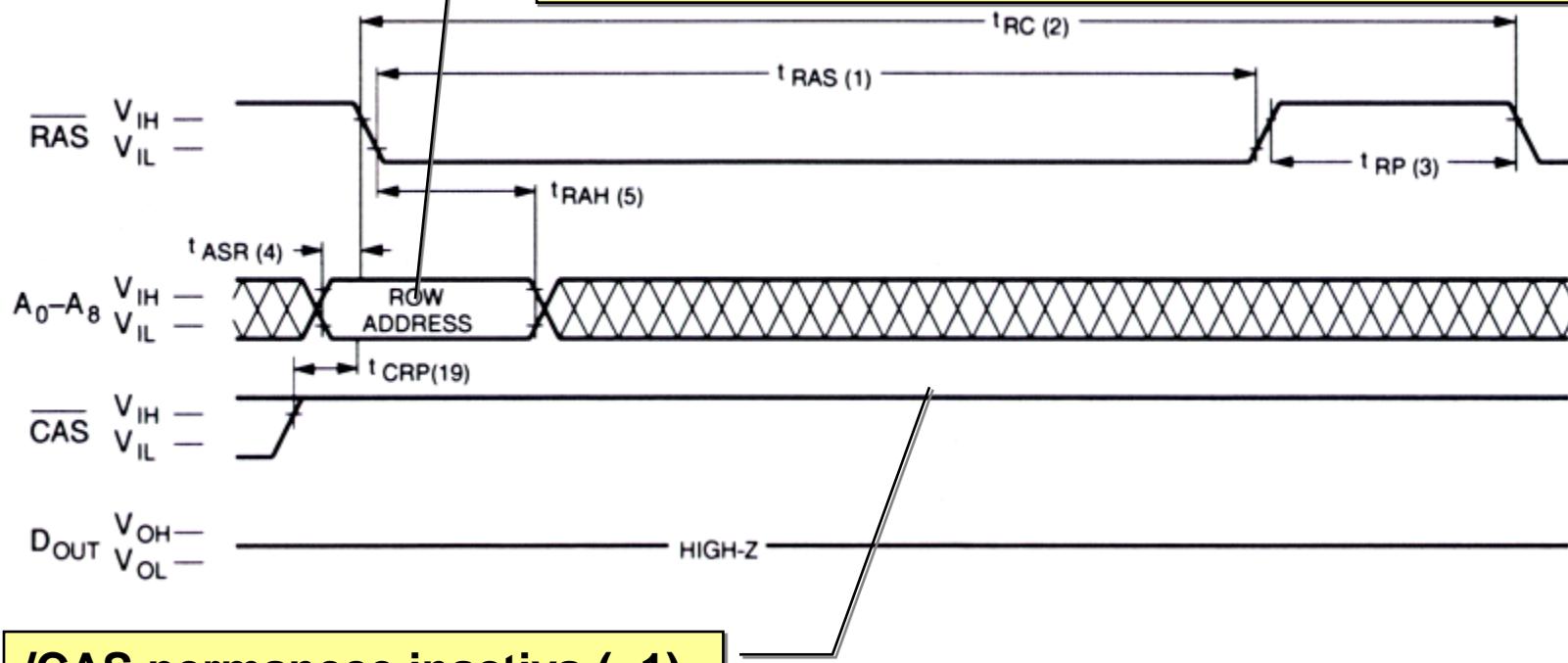
$$\frac{1024}{t_{RC} + 1023 \cdot t_{PC}}$$

Para $t_{RC} = 120$ ns y $t_{PC} = 40$ ns $\Rightarrow 25$ MHz

Ejemplo de DRAM

- Ciclo de refresco sólo-/RAS (/RAS-only):

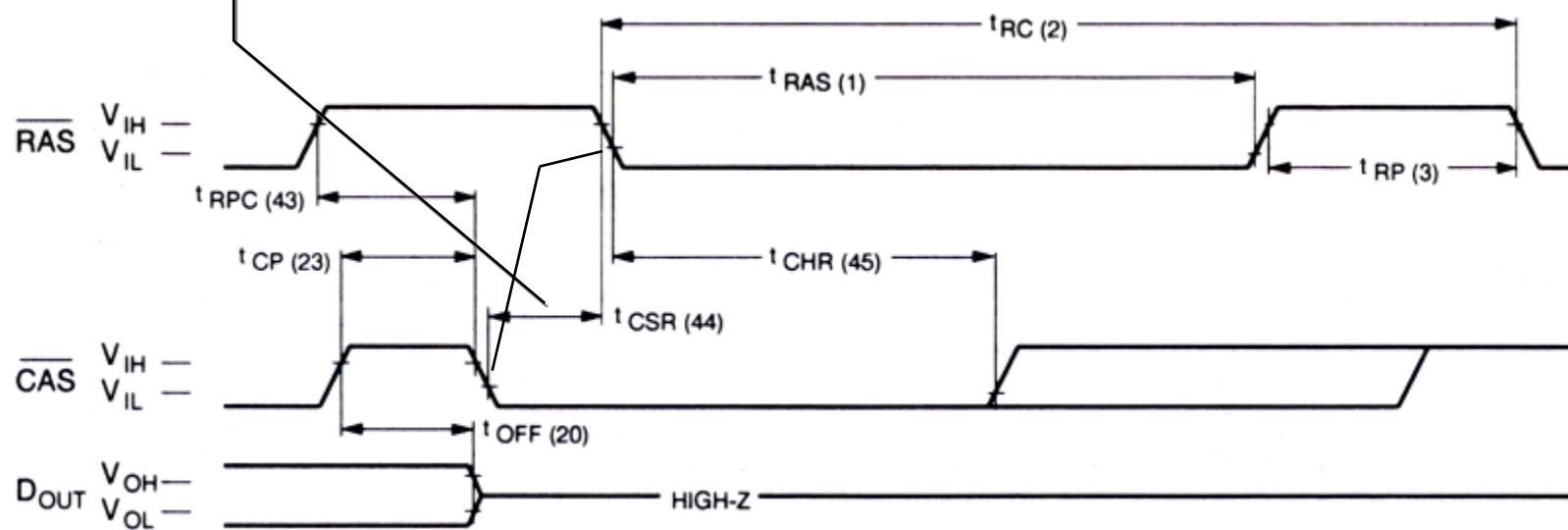
/RAS se activa ($=0$) \Rightarrow se valida la dirección de fila suministrada por un contador externo que va generando las direcciones de las filas a refrescar.



Ejemplo de DRAM

- Ciclo de refresco /CAS-antes de-/RAS (/CAS-before-/RAS):

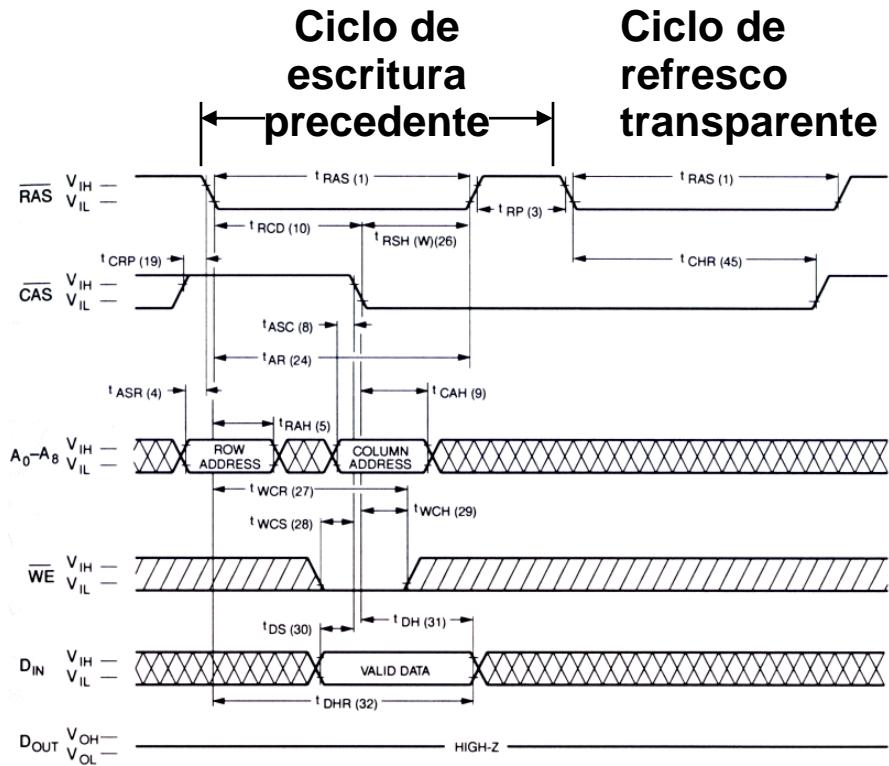
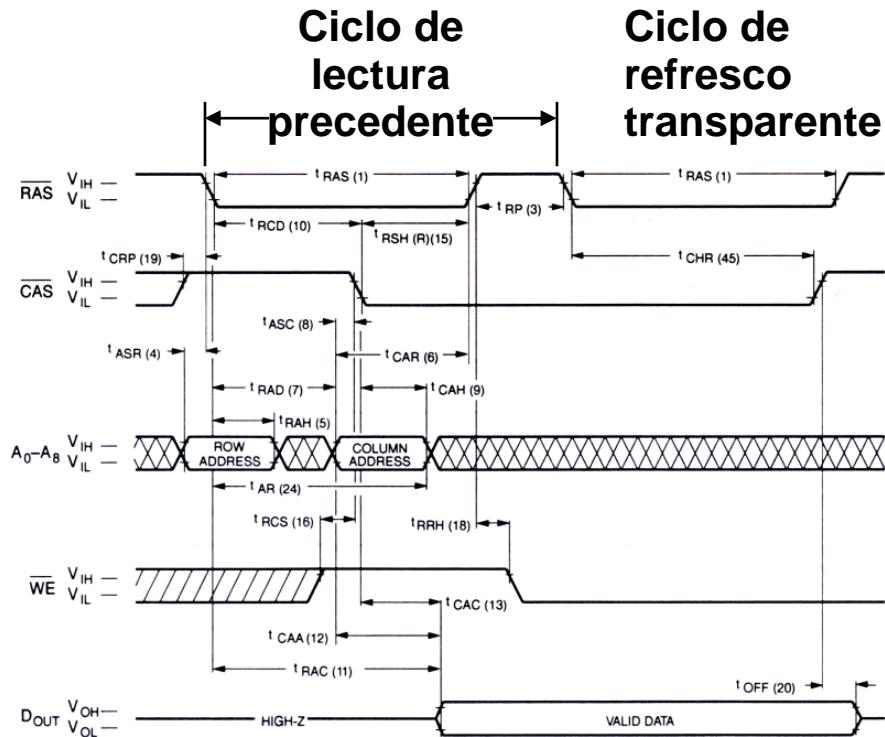
/CAS y /RAS se activan en orden inverso \Rightarrow se usa la salida del contador interno de 9 bits como fuente de la dirección de fila a refrescar, ignorándose las entradas de dirección externas.



Ejemplo de DRAM

- **Refresco transparente (Hidden refresh):**

- Similar a un ciclo /CAS-antes de-/RAS, pero en este caso /CAS no se pone a 1 y luego a 0, sino que permanece a 0 después del ciclo de lectura o escritura precedente.



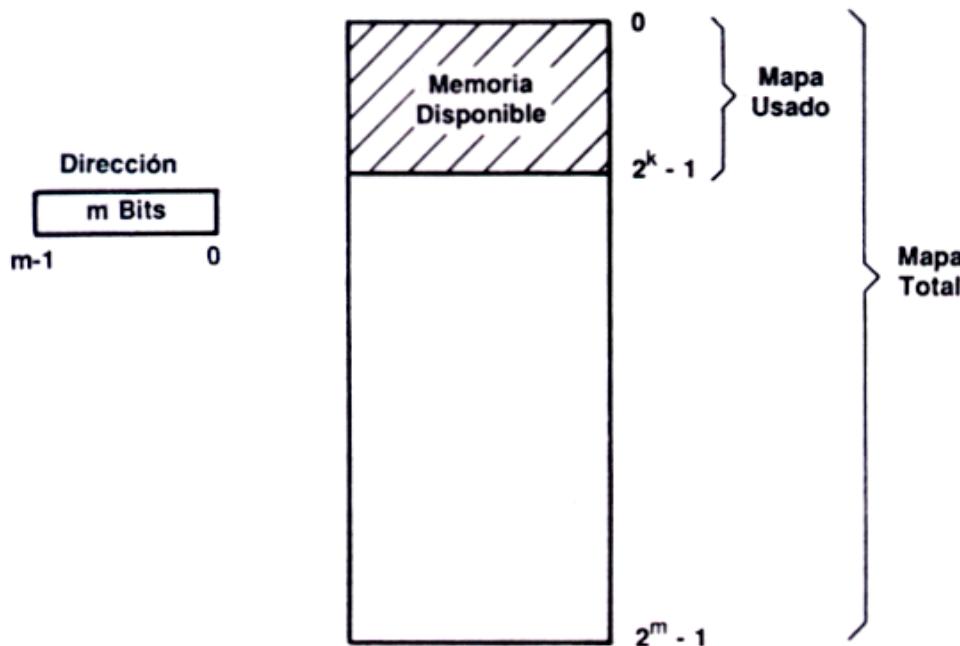
Memoria

- **Jerarquía de memoria. Concepto de localidad**
- **Memorias RAM semiconductoras. Memorias de sólo lectura. Prestaciones: velocidad, tamaño y coste**
- **Configuración y diseño de memorias utilizando varios chips**
- **Memorias asociativas**
- **Memoria cache. Influencia en las prestaciones**

Ampliación del mapa de memoria

■ Mapa de memoria

- Espacio direccionable por un procesador.
- Viene determinado por el tamaño de las direcciones.
- Generalmente un ordenador no dispone de toda la memoria necesaria para llenar el mapa de memoria, sino que queda espacio libre para posibles ampliaciones:

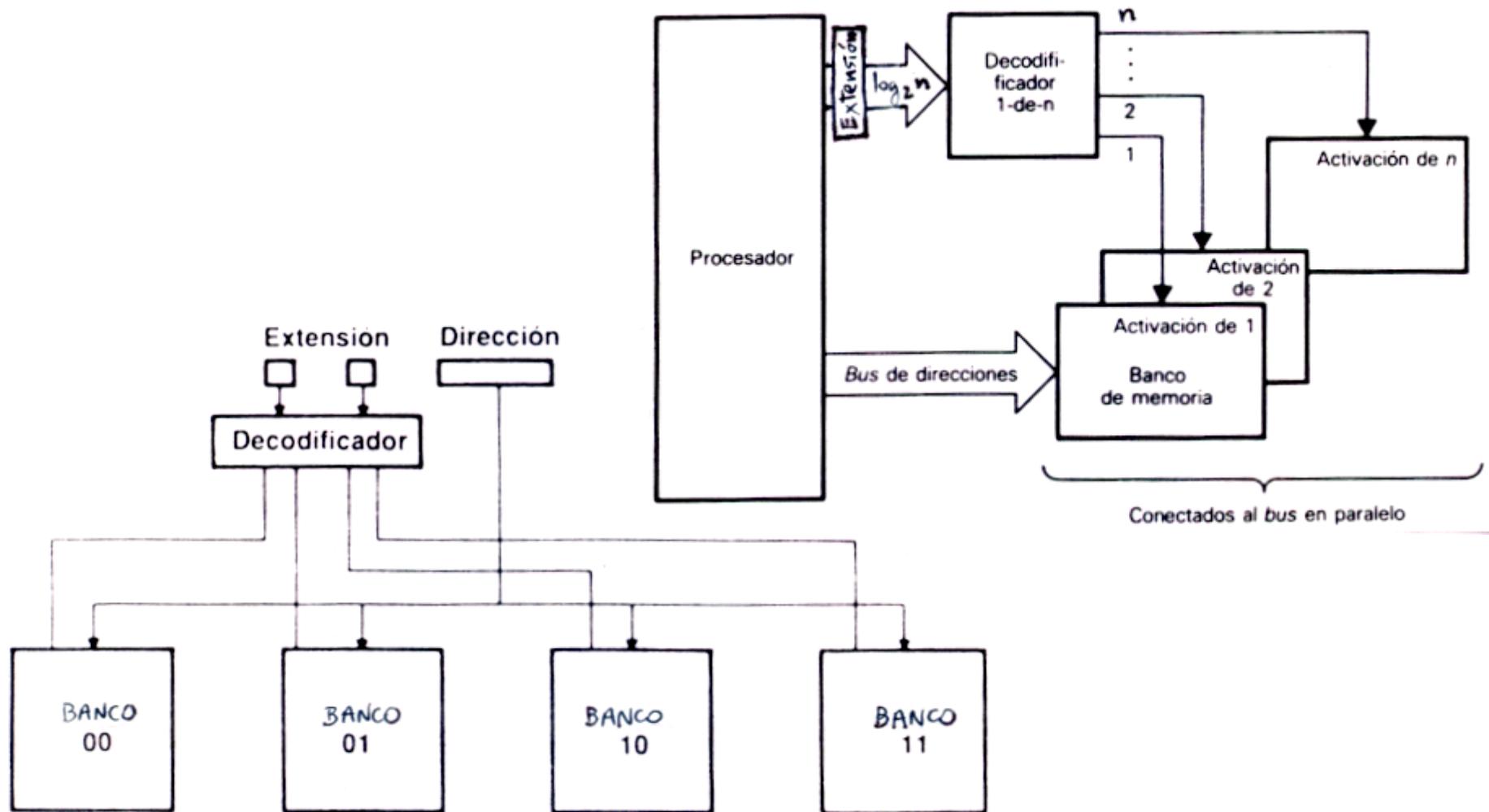


Ampliación del mapa de memoria

- Sin embargo, algunas arquitecturas con el tiempo se quedan pequeñas en cuanto a mapa de memoria.
- **Ampliación del mapa de memoria:**
 - ① Rediseñar el procesador para que sea capaz de generar direcciones con mayor número de bits.
 - ② Concatenar a las direcciones generadas por el procesador unos bits externos adicionales:
 - Se pueden modificar por programa (generalmente por medio de instrucciones de E/S).
 - Permiten seleccionar uno entre varios bancos de memoria distintos.
 - Este método es conocido como **CONMUTACIÓN DE BANCOS**.

Ampliación del mapa de memoria

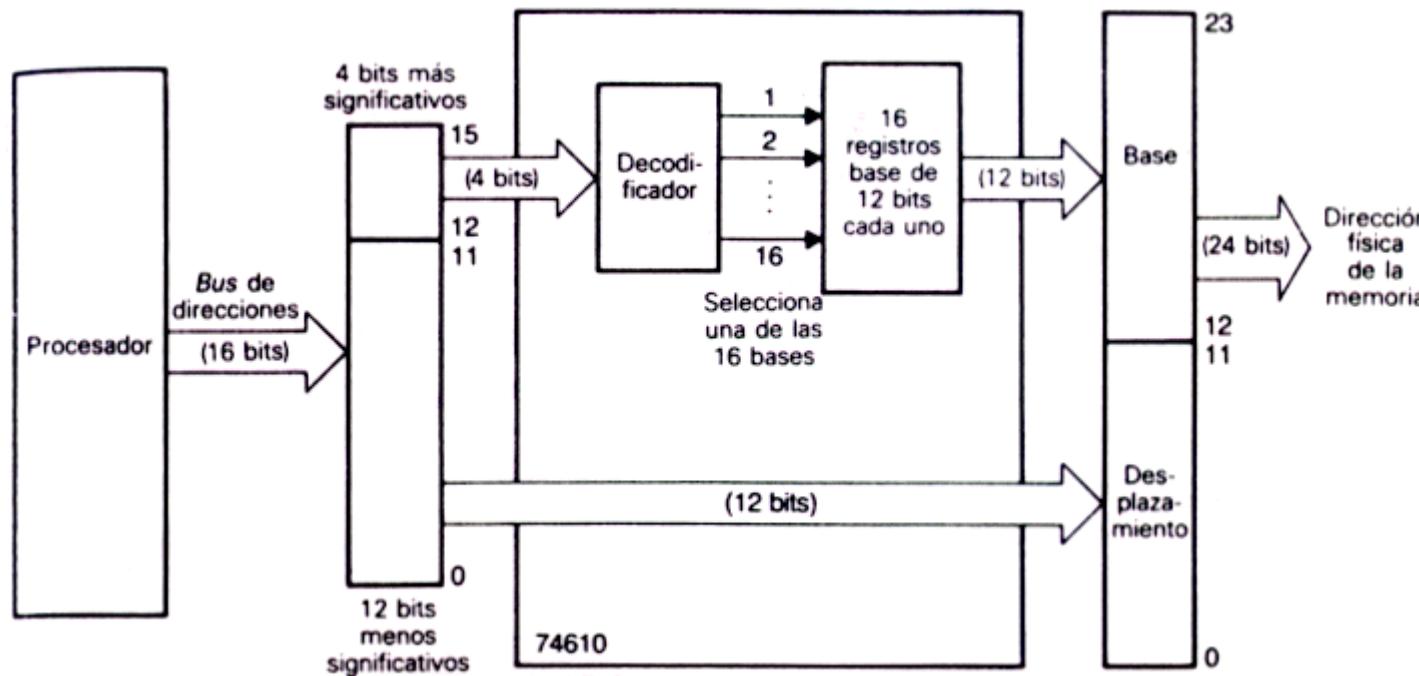
■ Conmutación de bancos:



Ampliación del mapa de memoria

- ③ Utilizar algunos de los bits más significativos del bus de direcciones para seleccionar un registro base de entre varios disponibles.
- Ejemplo:

- Chip 74610 de *Texas Instruments*.
- Con 16 bits de direcciones permite acceder a 16 MB de memoria física.



Ampliación de memoria

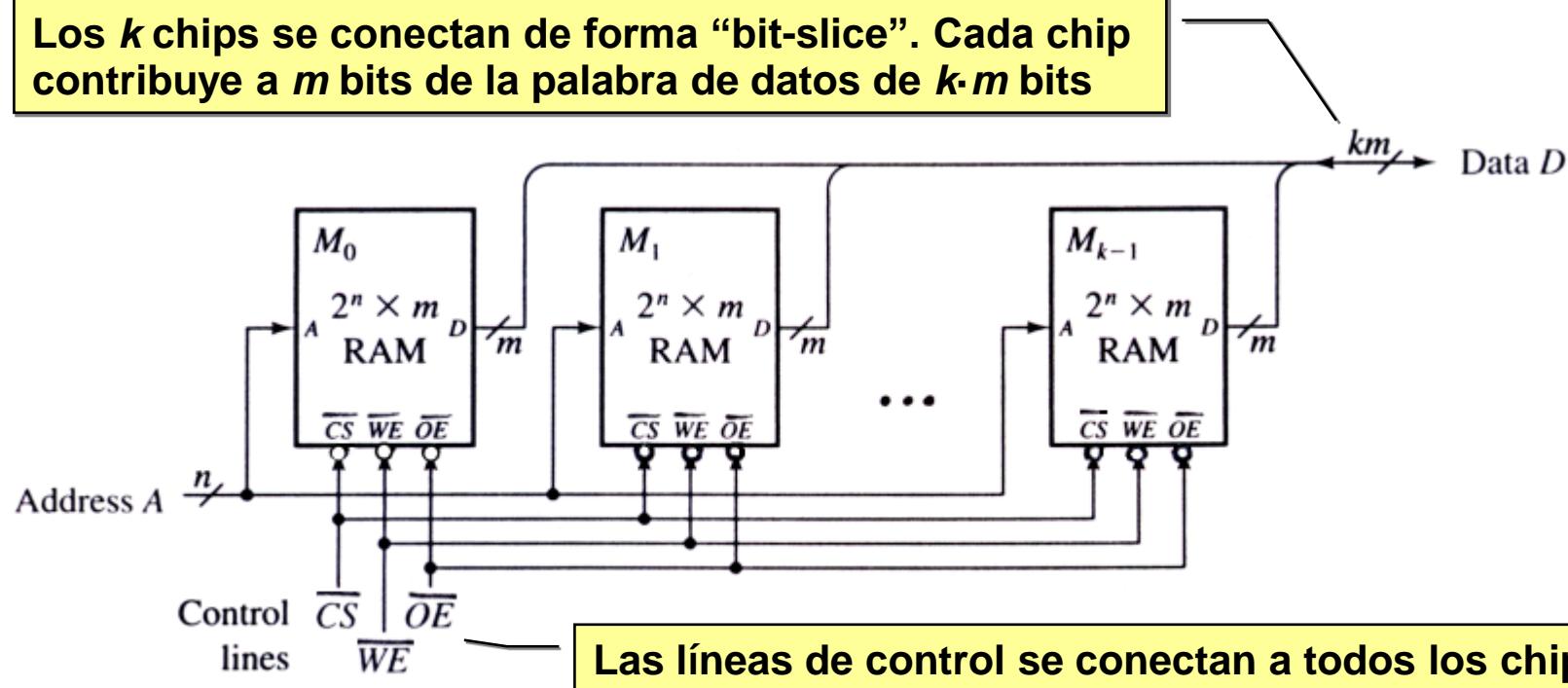
■ Problema:

- Construir una memoria de 2^N palabras de M bits a partir de chips de 2^n palabras de m bits.

① Incrementar el ancho de palabra de m a $k \cdot m = M$

- Necesitamos k circuitos de tamaño de palabra m :

Los k chips se conectan de forma “bit-slice”. Cada chip contribuye a m bits de la palabra de datos de $k \cdot m$ bits



Ampliación de memoria

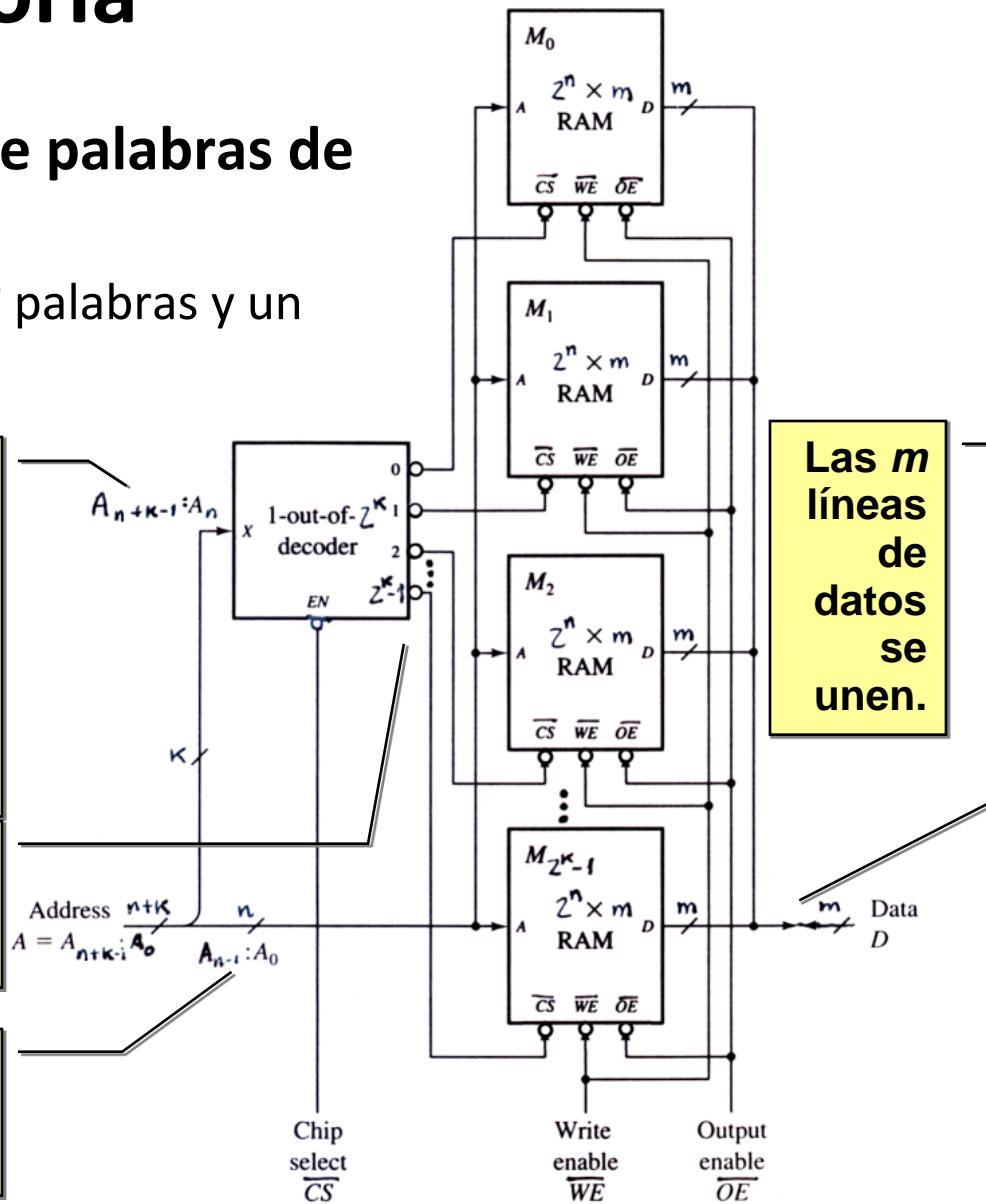
② Incrementar el número de palabras de 2^n a $2^{n+k}=2^N$

- Necesitamos 2^k circuitos de 2^n palabras y un decodificador de 1 entre 2^k .

Las k líneas de dirección de orden superior ($A_{n+k-1}:A_n$) se conectan a las entradas de un decodificador de k a 2^k \Rightarrow 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 2^k líneas de salida del decodificador se conectan a las entradas de selección /CS de los 2^k circuitos.

Las n líneas de dirección de orden inferior se conectan en común a las líneas de dirección de los 2^k chips.



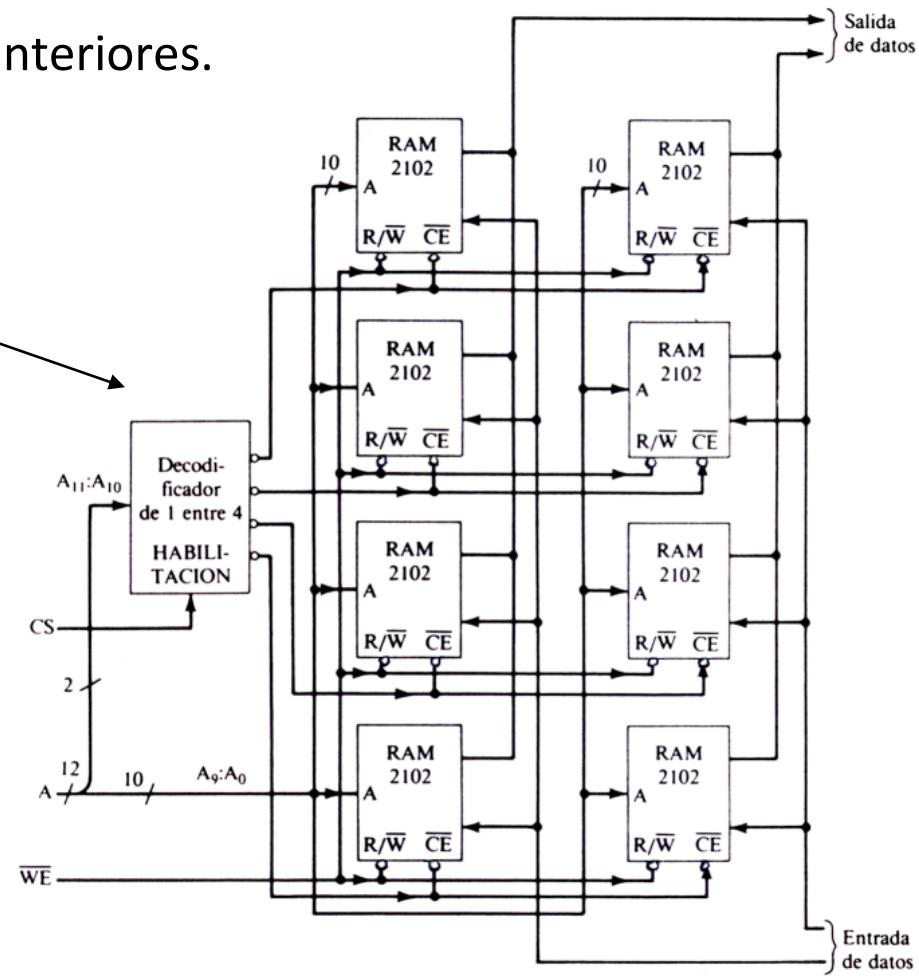
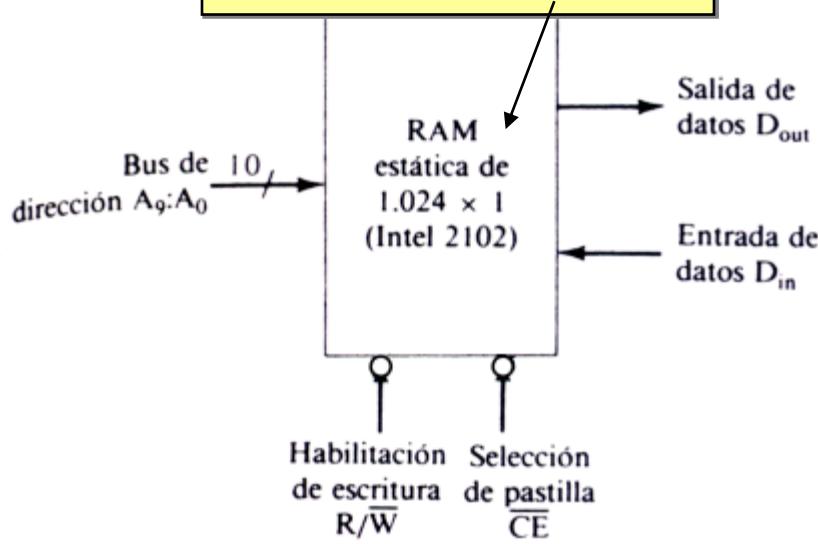
Ampliación de memoria

③ Incrementar simultáneamente el número de palabras y el ancho de palabra.

- Basta combinar las dos técnicas anteriores.

Ejemplo 1:

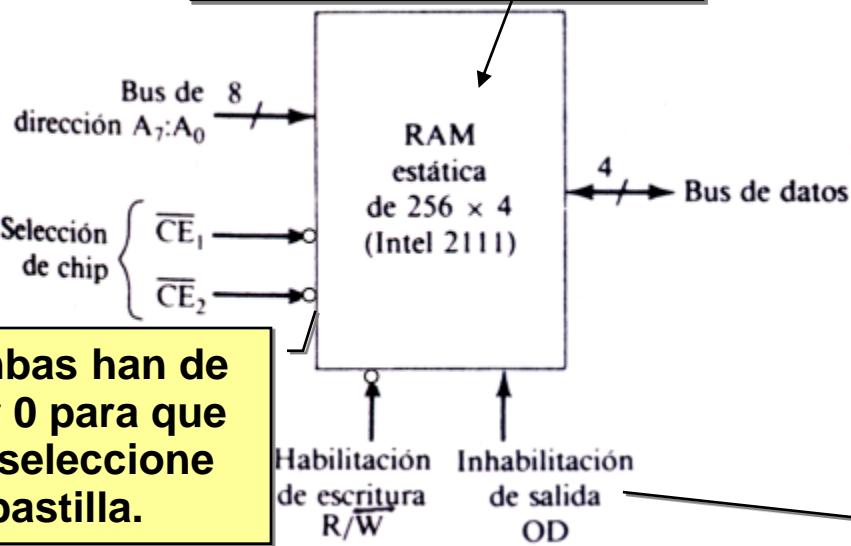
RAM de 4096×2 construida con 8 RAM de 1024×1 .



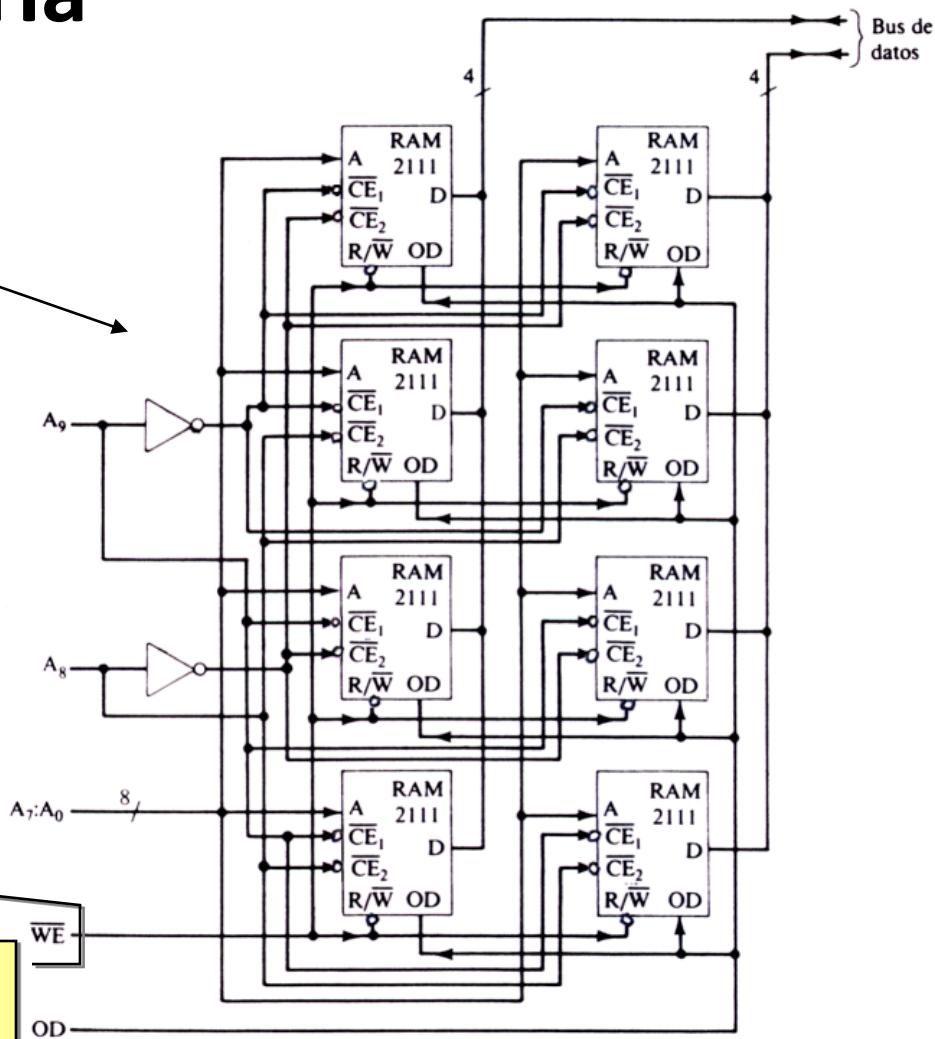
Ampliación de memoria

Ejemplo 2:

**RAM de $1\text{ K} \times 8$
construida con 8
RAM de 256×4 .**



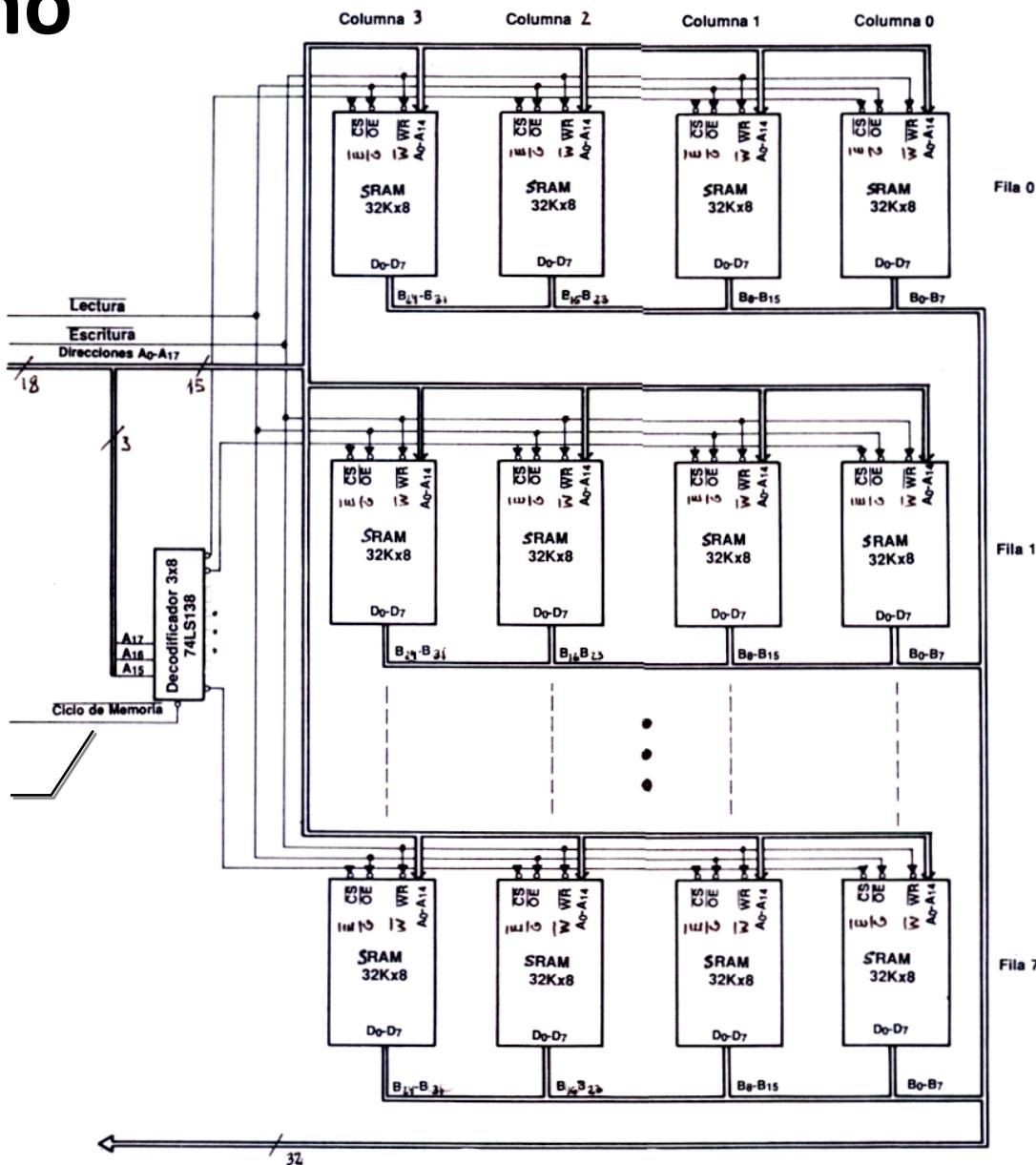
Como hay un solo bus de datos \Rightarrow la línea **OD (Output Disable)** se activa durante los ciclos de escritura para evitar que se lean datos internos desde el chip al bus mientras éste se está utilizando como bus de entrada.



Ejemplos de diseño

- Ejemplo 1:
 - SRAM de $256 \text{ K} \times 32$ empleando pastillas de $32 \text{ K} \times 8$.
 - Se requieren 8 filas de 4 pastillas = 32 pastillas.

Si /Ciclo_de_Memoria = 1 no se activa ninguna fila de pastillas.



Ejemplos de diseño

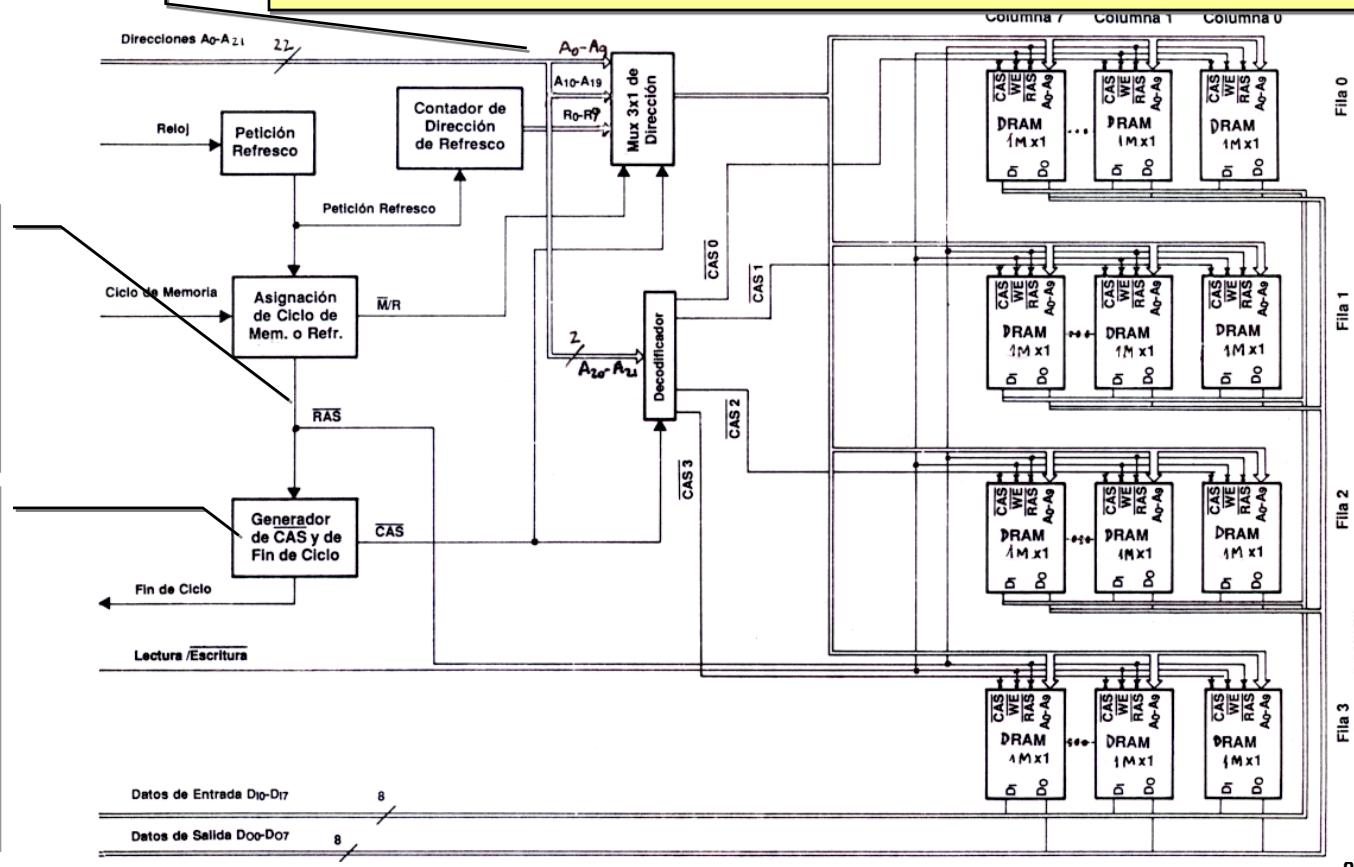
- Ejemplo 2:
 - DRAM de 4 Mbytes empleando pastillas de 1 M × 1.

/M / R	/CAS	Direcciones
0	0	A ₀ -A ₉
0	1	A ₁₀ -A ₁₉
1	X	R ₀ -R ₉

La señal /RAS especifica la primera mitad de la dirección y es común a todas las pastillas.

La señal /CAS, además de especificar la otra mitad, sirve como /CS y se activa de forma independiente para cada fila de pastillas.

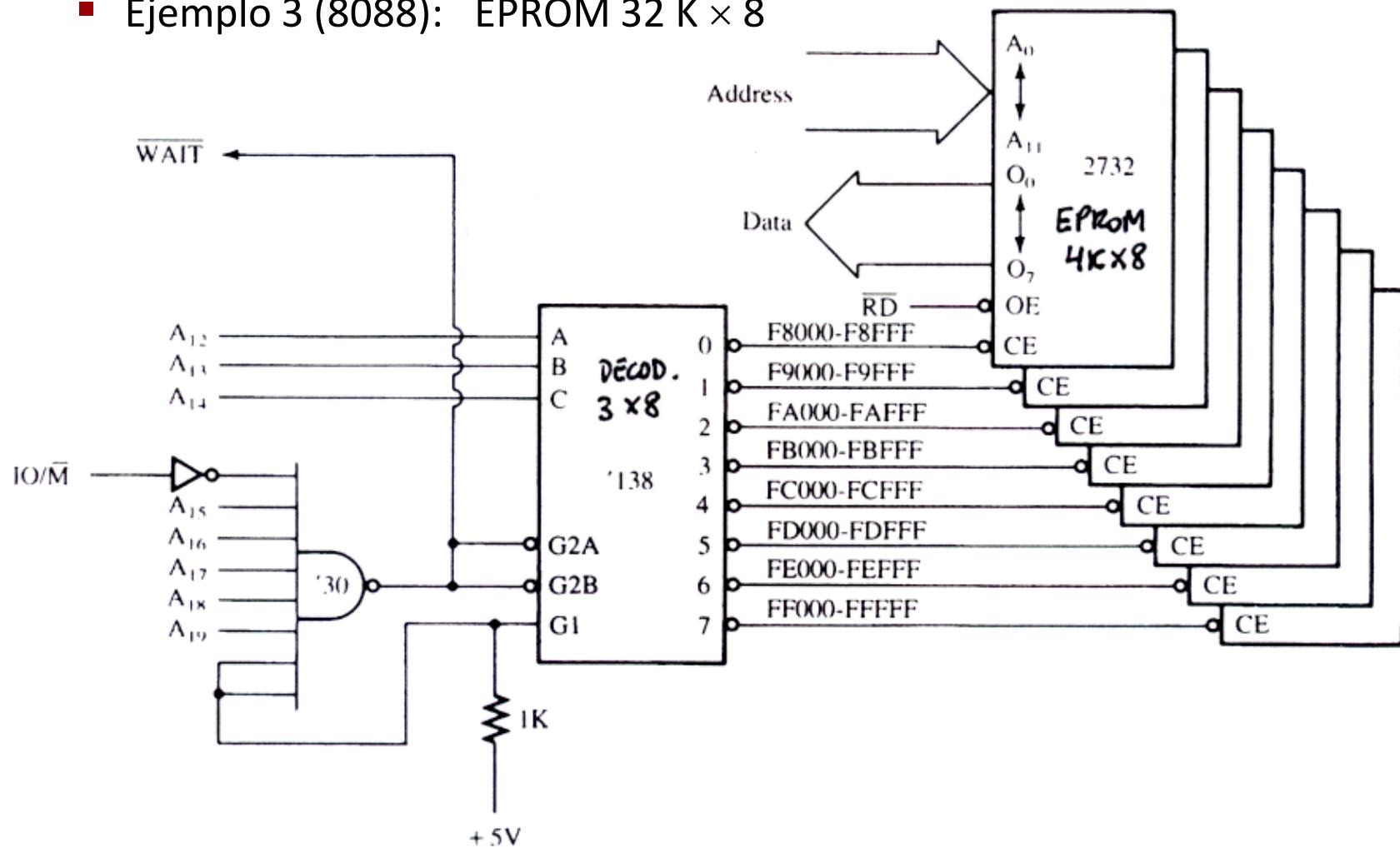
El multiplexor permite enviar a las pastillas la parte inferior o superior de los 16 bits de dirección.



Ejemplos de diseño

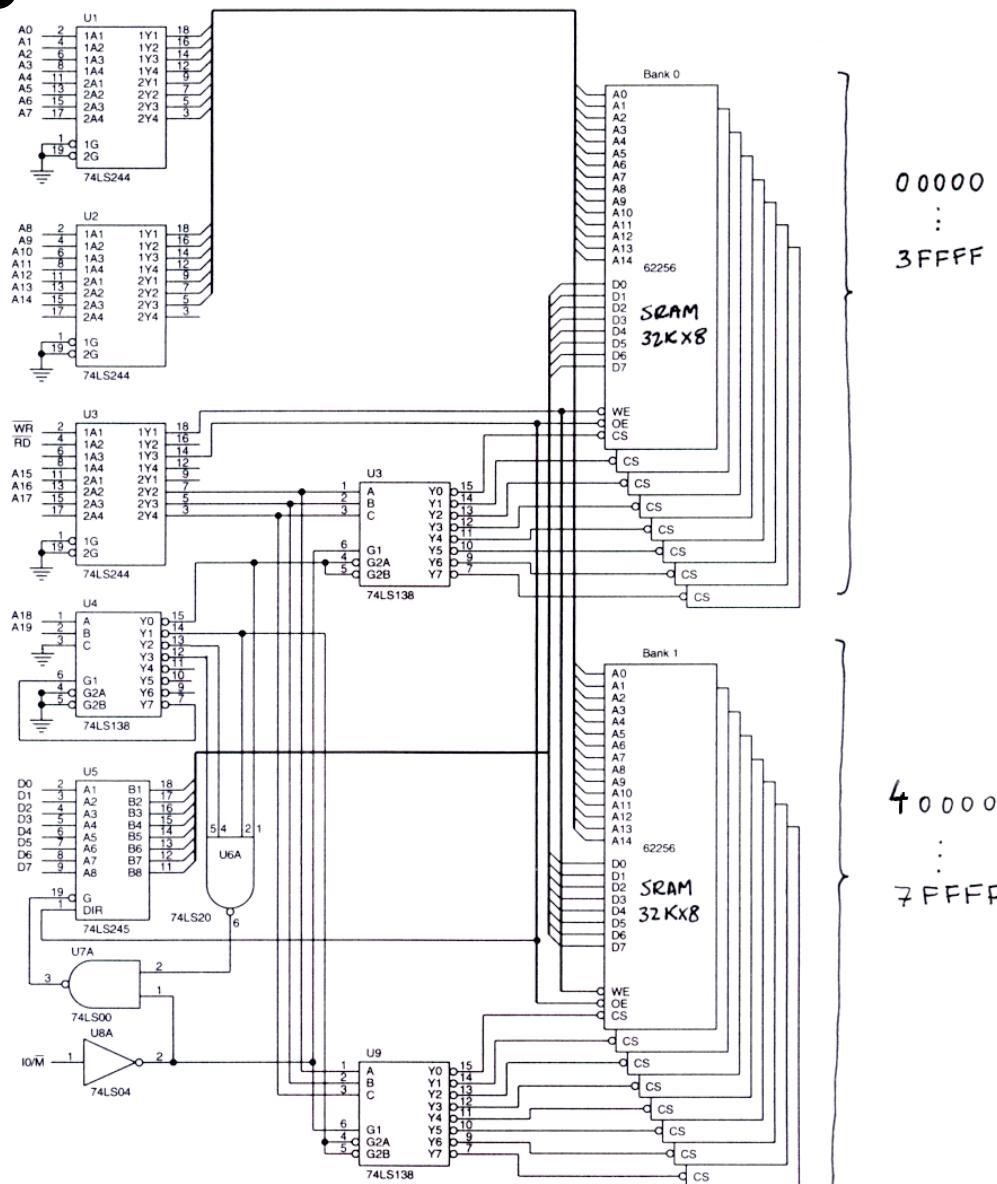
■ Configuraciones para μprocesadores Intel:

- Ejemplo 3 (8088): EPROM 32 K × 8



Ejemplos de diseño

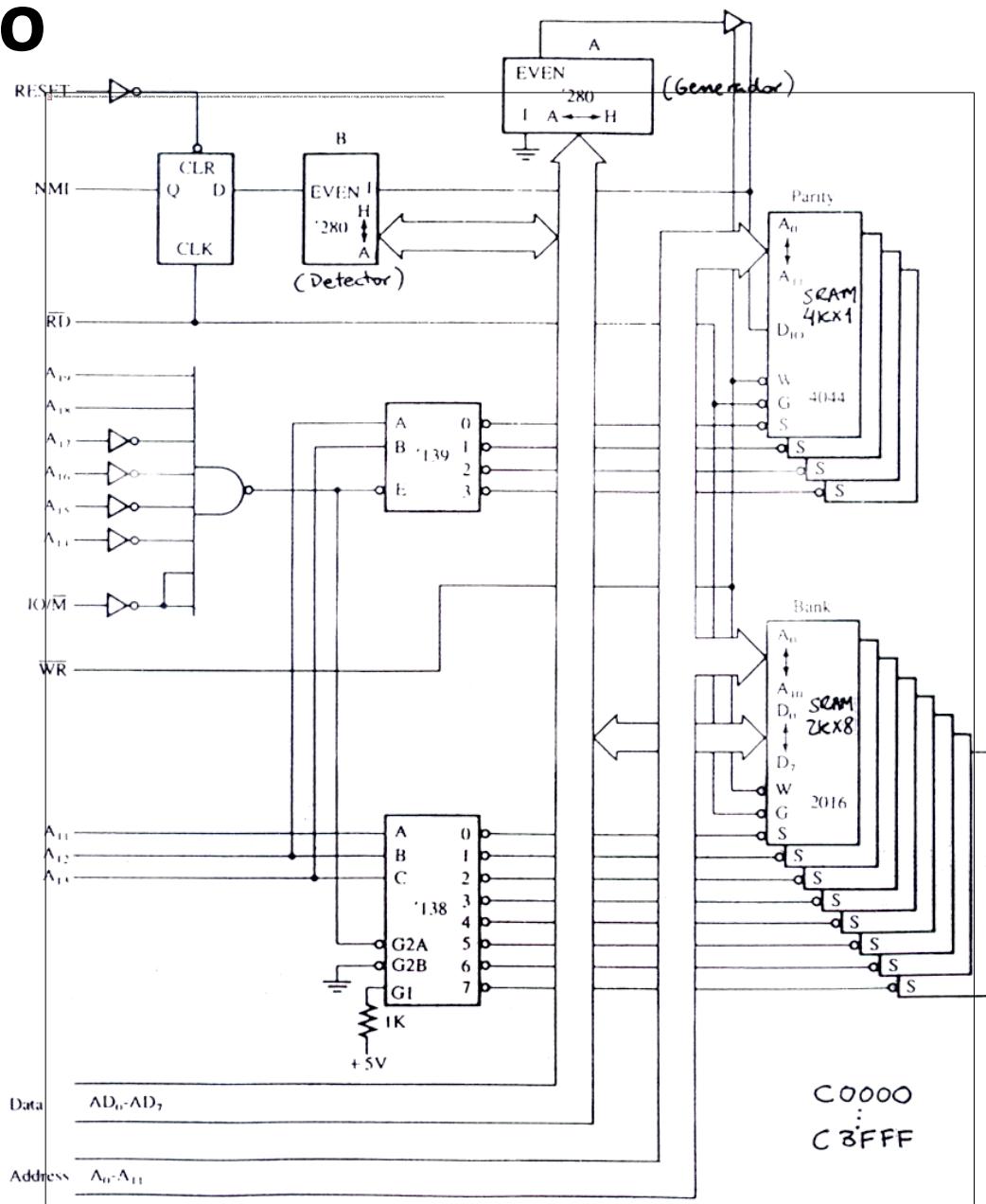
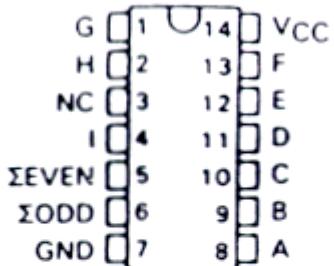
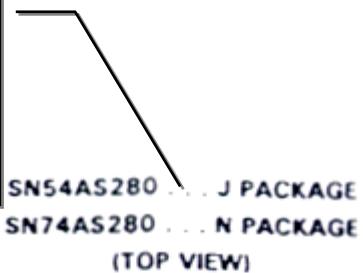
- Ejemplo 4 (8088):
SRAM 512 K × 8



Ejemplos de diseño

- Ejemplo 5 (8088):
 - SRAM 16 K × 8
 - Con paridad impar.

Circuito generador / detector de paridad.



C0000
:
C8FFF

Ejemplos de diseño

- Organización de memoria del 8086 / 286 / 386SX.

286/386SX (8MB)



286/386SX (8MB)

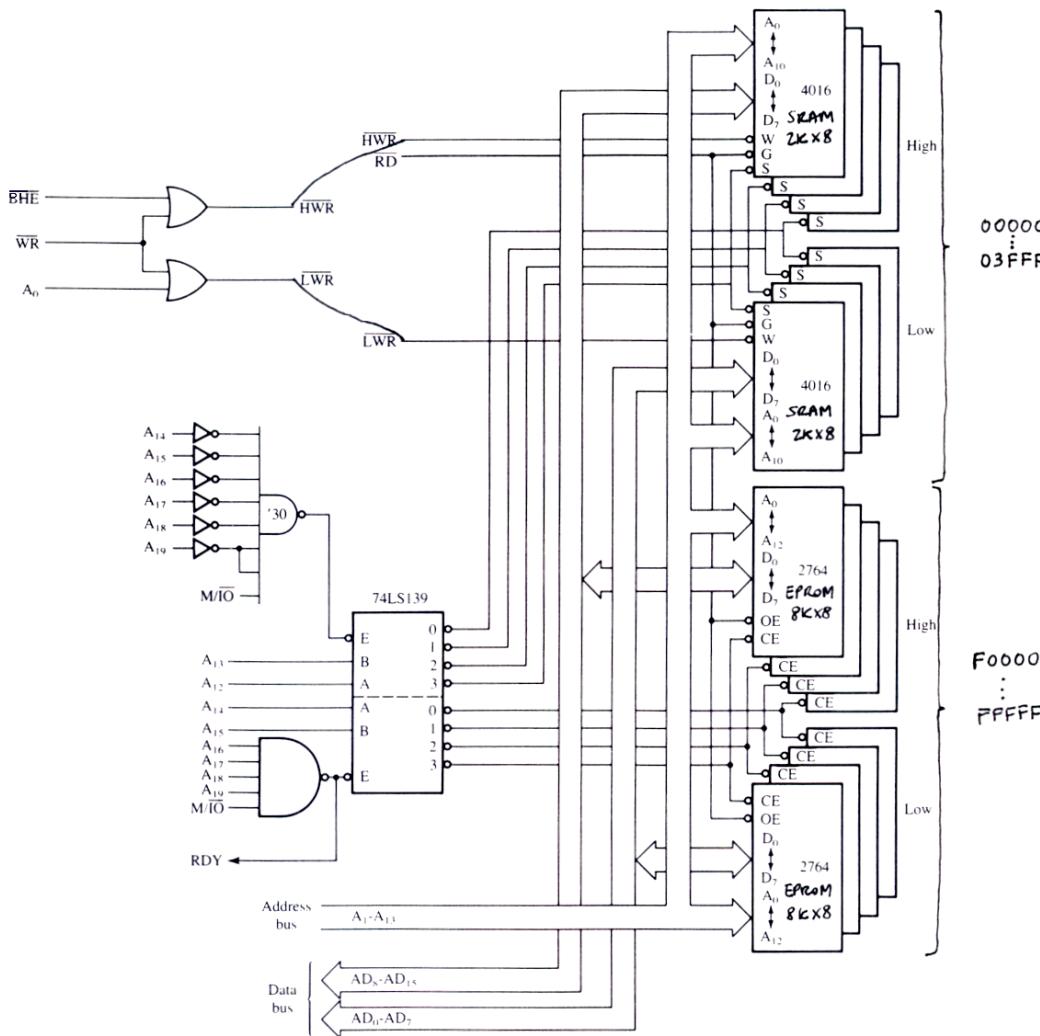


Note: A_0 is labeled \overline{BLE} (BUS Low enable) on the 80386SX.

\overline{BHE}	A_0	Function
0	0	Both banks active (16-bit transfer through $D_{15}-D_0$)
0	1	High bank active (8-bit transfer through $D_{15}-D_8$)
1	0	Low bank active (8-bit transfer through D_7-D_0)
1	1	No banks active

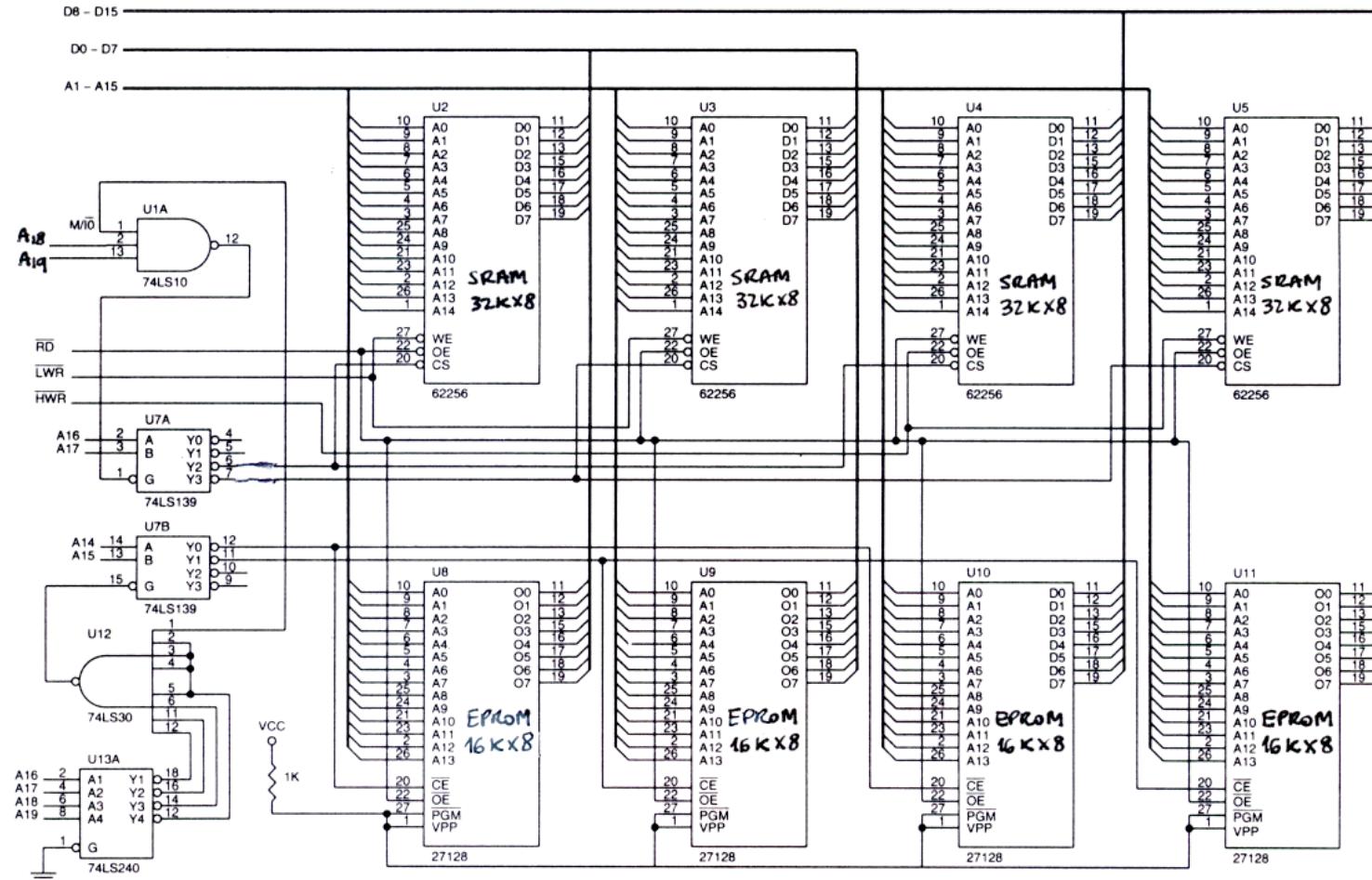
Ejemplos de diseño

- Ejemplo 6 (8086) SRAM 8 K × 16 + EPROM 32 K × 16



Ejemplos de diseño

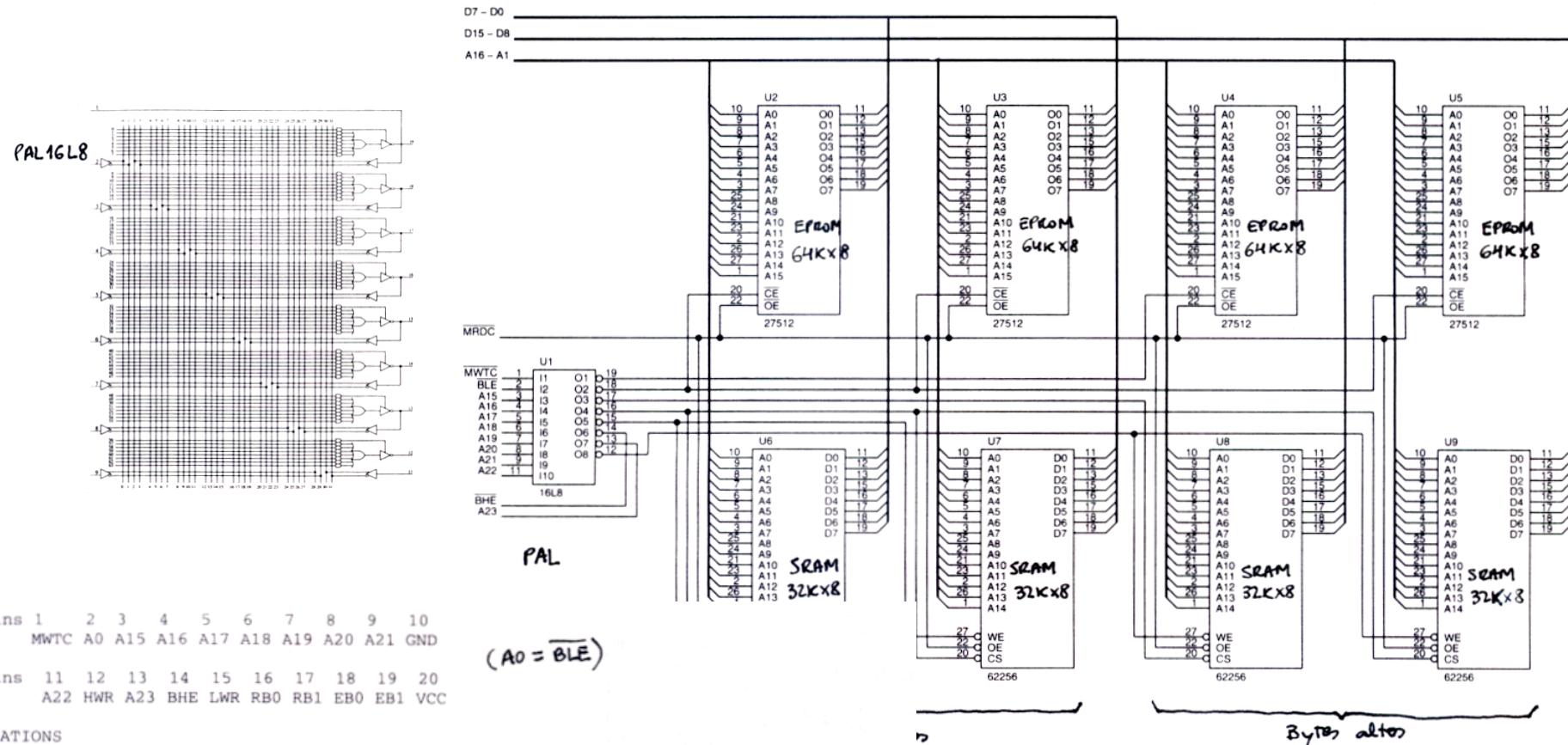
- Ejemplo 7 (8086) SRAM 64 K × 16 + EPROM 32 K × 16



EPROM: 00000-0FFFF
SRAM: E0000-FFFFF

Ejemplos de diseño

- Ejemplo 8 (386SX) EPROM 128 K × 16 + SRAM 64 K × 16



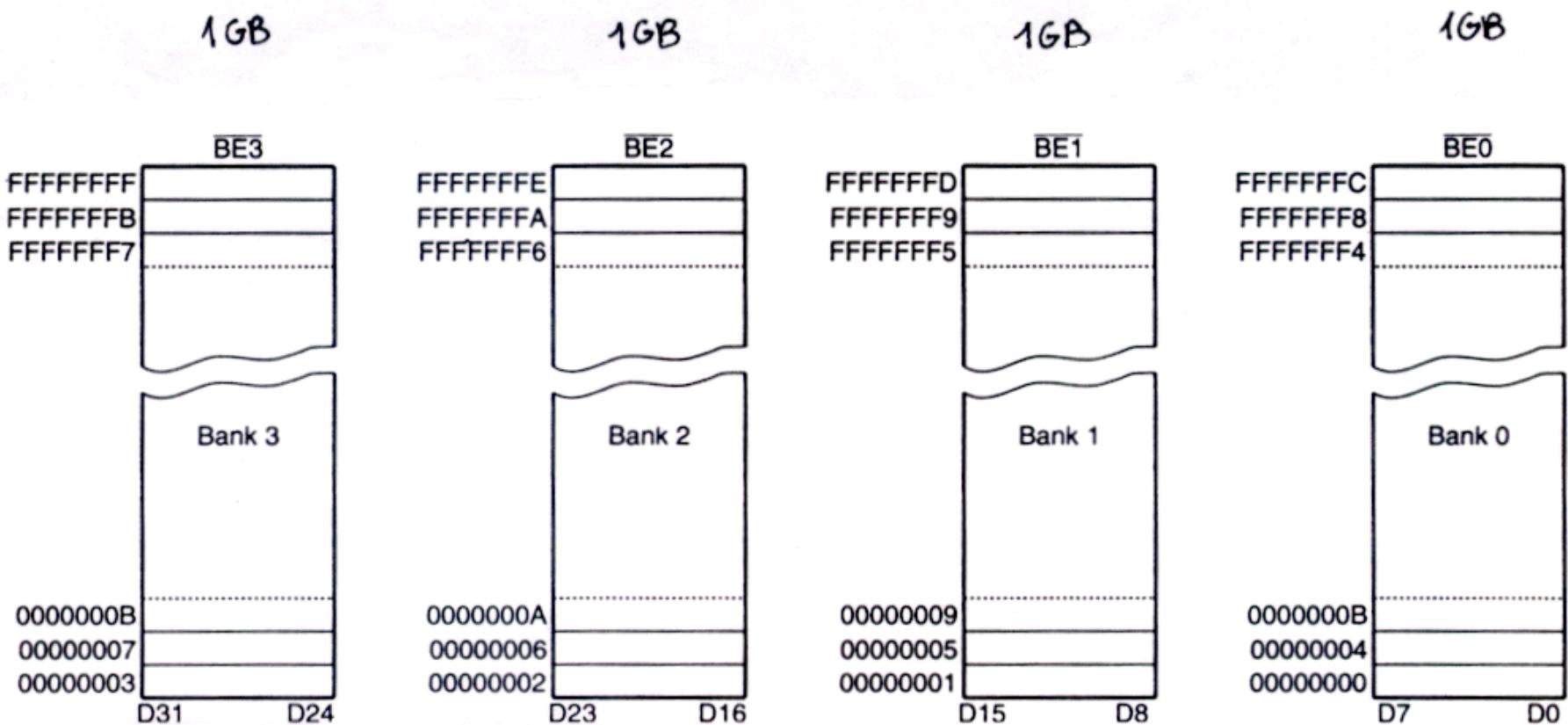
} EPROM :
 FC0000 :
 FFFFFF

00 0000
 :
 01 FFFF

Bytes altos

Ejemplos de diseño

- Organización de memoria del 386DX / 486.



Ejemplos de diseño

- Ejemplo 9 (386 DX / 486) SRAM 64 K × 32

CHIP Decoder1U1 PAL16L8
;pins 1 2 3 4 5 6 7 8 9 10
MWTC BE0 BE1 BE2 BE3 A17 A28 A29 A30 GND
;pins 11 12 13 14 15 16 17 18 19 20
A31 RB1 U2 NC WR0 WR1 WR2 WR3 RB0 VCC

EQUATIONS

$$\begin{aligned} \text{/WR0} &= \text{/MWTC} * \text{/BE0} \\ \text{/WR1} &= \text{/MWTC} * \text{/BE1} \\ \text{/WR2} &= \text{/MWTC} * \text{/BE2} \\ \text{/WR3} &= \text{/MWTC} * \text{/BE3} \\ \text{/RB0} &= \text{/A31} * \text{/A32} * \text{/A30} * \text{/A29} * \text{/A28} * \text{/A17} * \text{/U2} \\ \text{/RB1} &= \text{/A31} * \text{/A32} * \text{/A30} * \text{/A29} * \text{/A28} * \text{/A17} * \text{/U2} \end{aligned}$$

0 0 0 0 0 0 0/4

CHIP Decoder1U2 PAL16L8

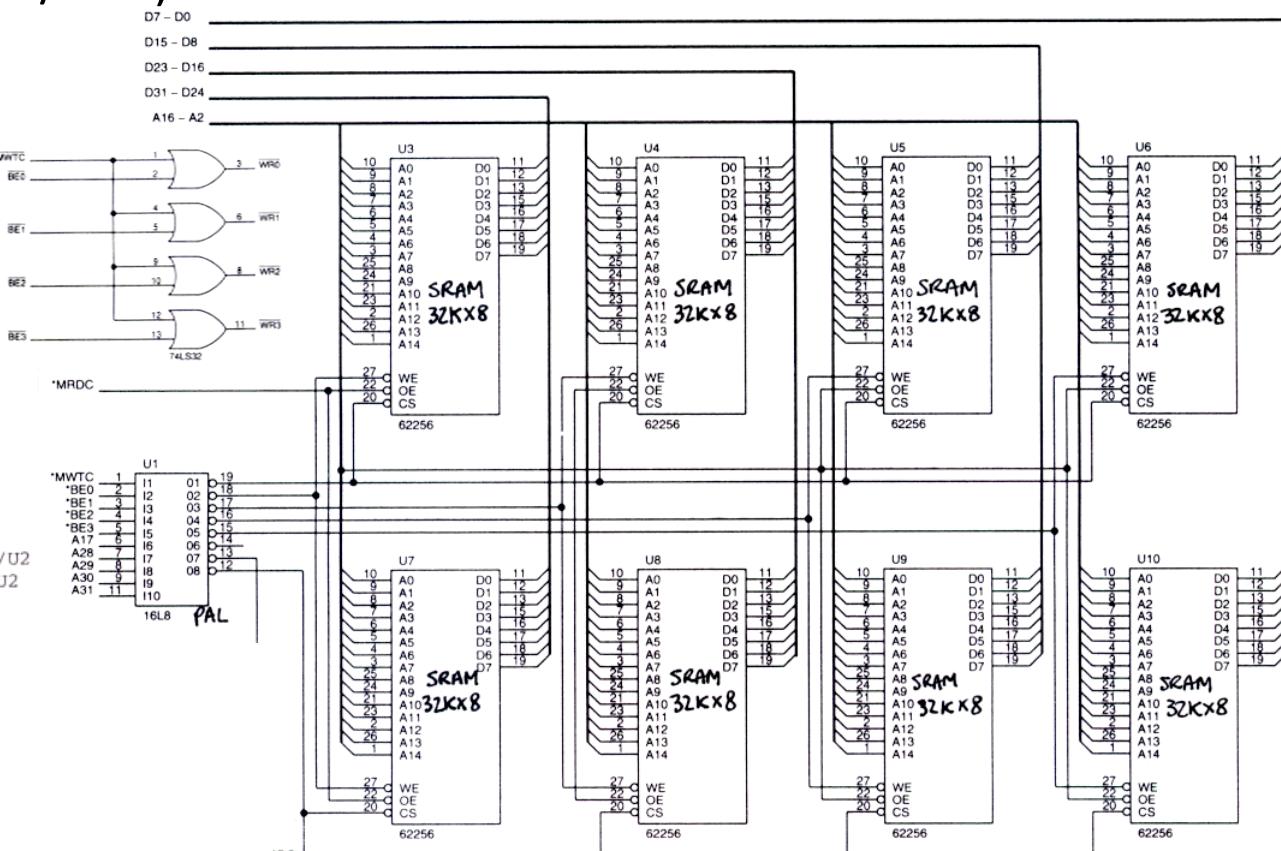
;pins 1 2 3 4 5 6 7 8 9 10
A18 A19 A20 A21 A22 A23 A24 A25 A26 GND
;pins 11 12 13 14 15 16 17 18 19 20
A27 U2 NC NC NC NC NC NC NC VCC

EQUATIONS

$$\text{/U2} = \text{/A27} * \text{/A26} * \text{/A25} * \text{/A24} * \text{/A23} * \text{/A22} * \text{/A21} * \text{/A20} * \text{/A19} * \text{/A18}$$

0 0 1 0 0 0 0 0 0

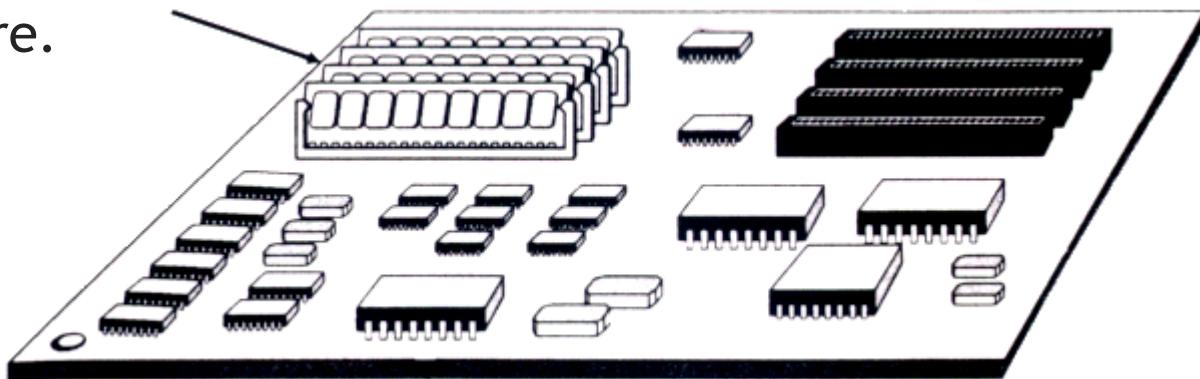
0200 0000
: Range de direcciones
0203 FFFF



Módulos de memoria en línea

- En los 80, la memoria solía soldarse directamente en la placa madre del ordenador. Pero a medida que aumentaron los requisitos de memoria, esta técnica resultó poco factible.
- SIMM, DIMM, SODIMM, RIMM, SORIMM:

- Varios chips DRAM en una pequeña placa de circuito impreso (PCB) que calza en un conector en la placa madre.



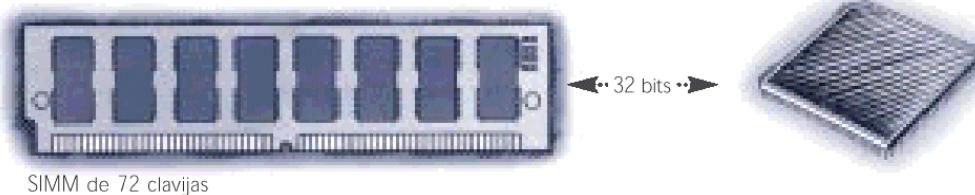
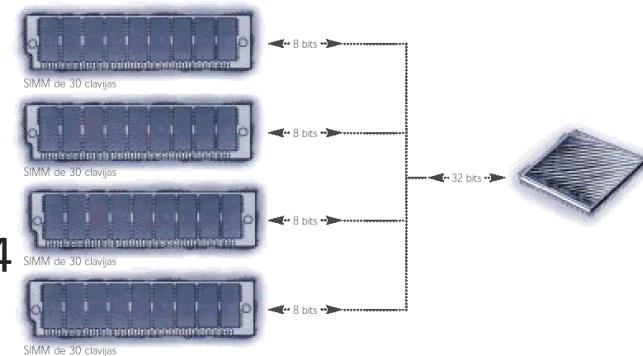
- ✓ método flexible para actualizar la memoria
- ✓ ocupa menos espacio en la placa madre.
- Normalmente sólo se amplía el bus de datos, no el nº de direcciones (no hay necesidad de decodificador).

Módulos de memoria SIMM



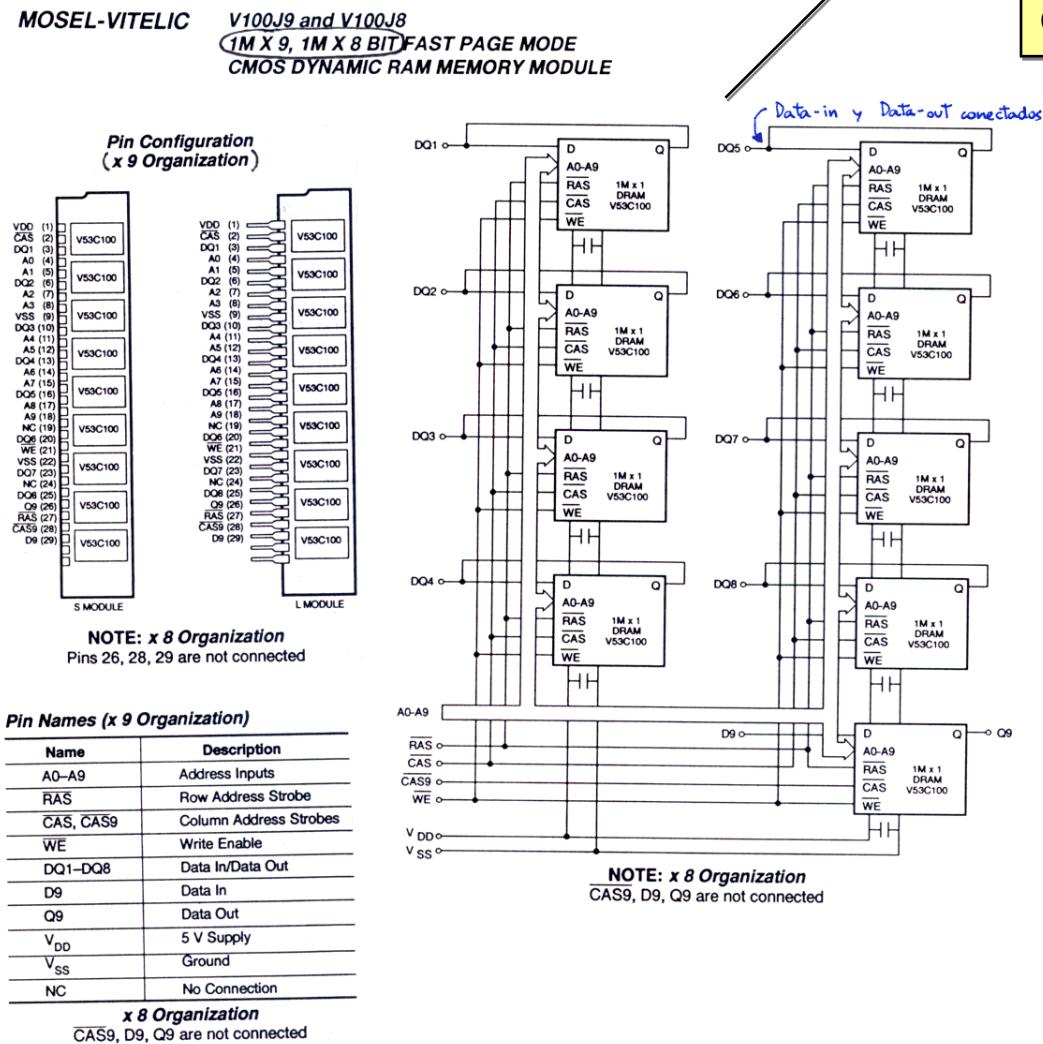
■ **SIMM (Single In-line Memory Module)**

- FPM y EDO
- 30 contactos:
 - 8 bits de datos
 - Si la placa madre tiene conectores para SIMM de 30 contactos \Rightarrow se necesitarán 4 SIMM para obtener 32 bits.
 - En un sistema de esta clase, la configuración de la memoria típicamente se divide entre dos bancos de memoria: el banco cero y el banco uno. Cada banco de memoria consiste en cuatro conectores de SIMM de 30 contactos. La CPU se dirige a un banco de memoria a la vez.
- 72 contactos:
 - 32 bits de datos.
 - Un único SIMM por banco.



Módulos de memoria SIMM

- Ejemplo de SIMM de 30 contactos:

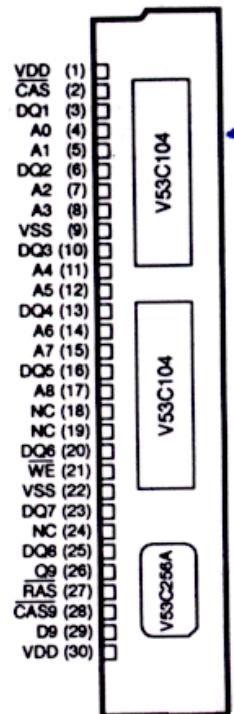


Módulos de memoria SIMM

- Más ejemplos de SIMM de 30 contactos:

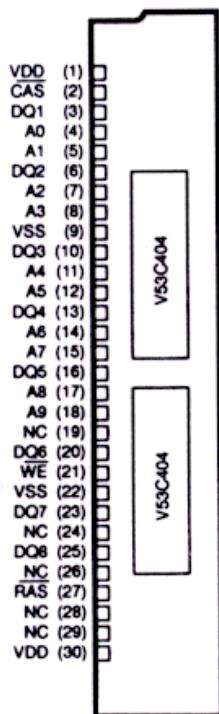
256KB

256 K X 8
256 K X 9



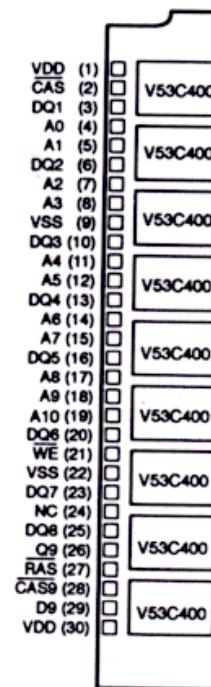
1MB

1 M X 8
1 M X 9



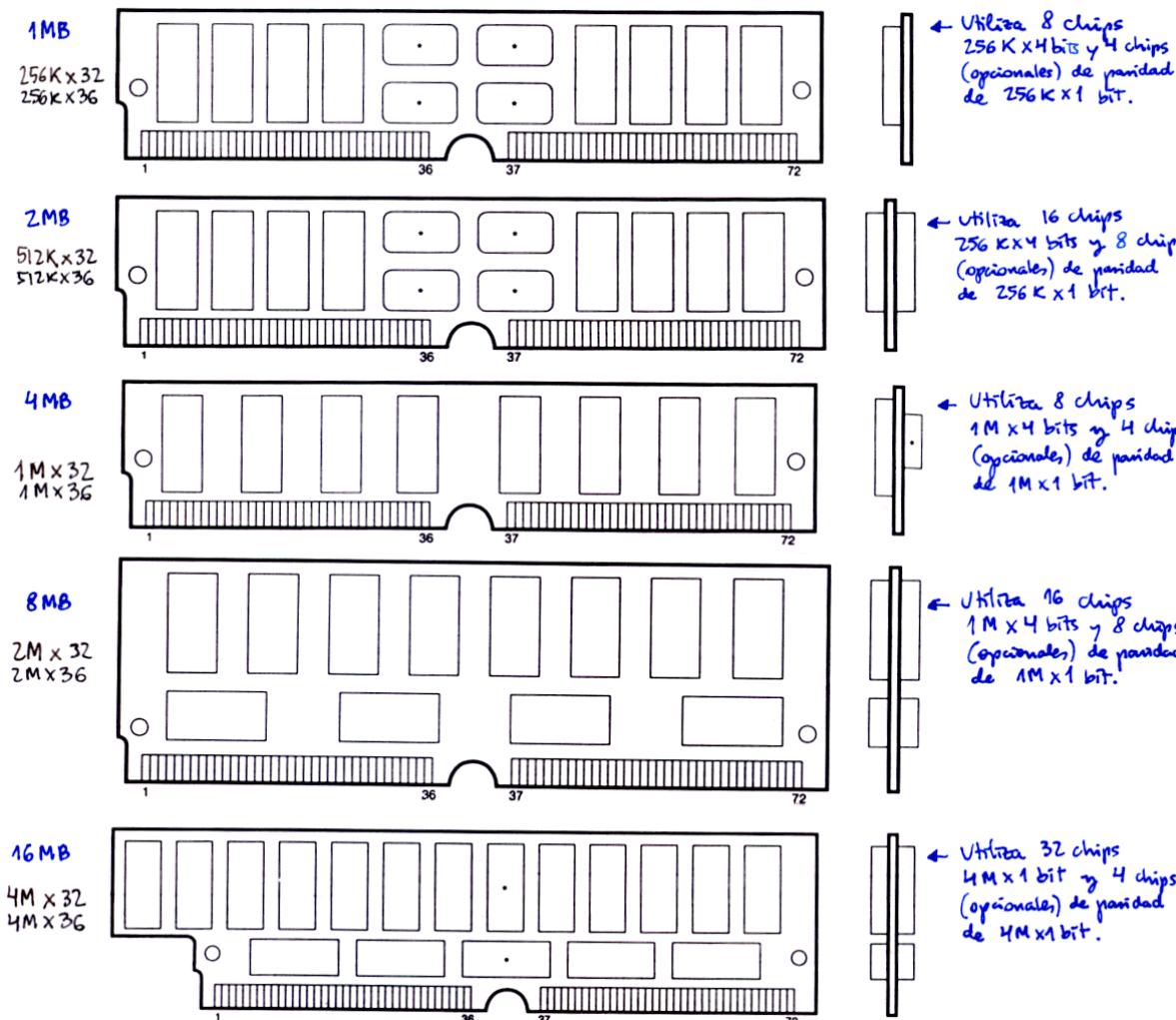
4MB

4 M X 8
4 M X 9



Módulos de memoria SIMM

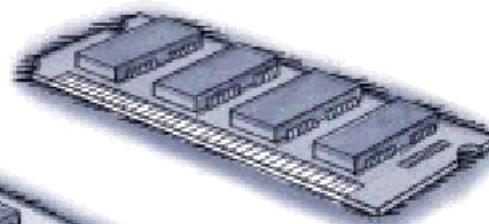
- Ejemplos de SIMM de 72 contactos:



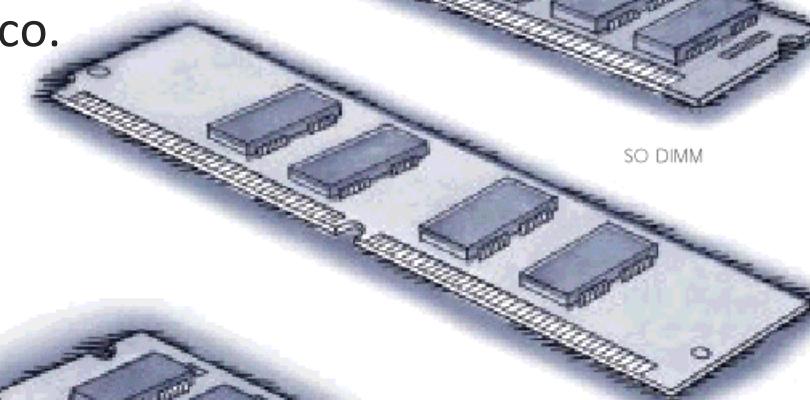
Módulos de memoria DIMM

■ **DIMM (Dual In-line Memory Module)**

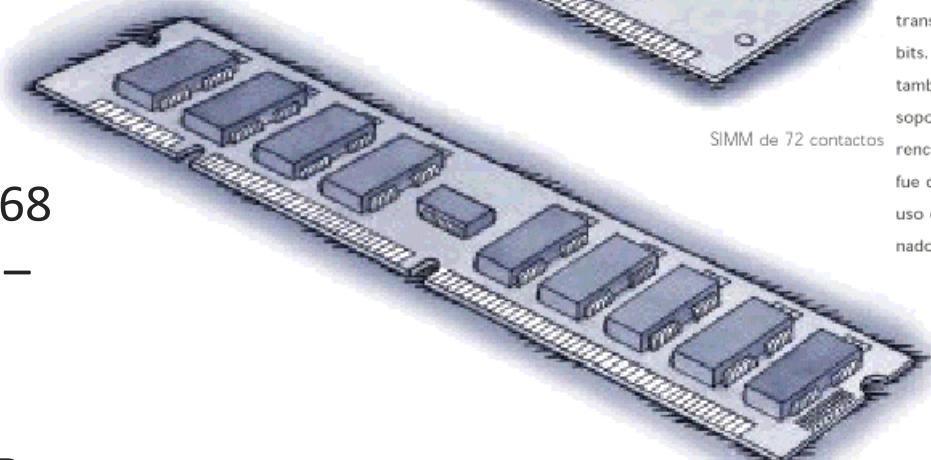
- En un SIMM, los contactos de cada fila se unen con los de la otra fila para formar uno único.
- En DIMM los contactos opuestos están aislados eléctricamente para formar **dos contactos separados**.
- 64 bits (ó 72 con ECC)
- Dos tipos de DIMM:
 - **FPM, EDO y SDRAM**, 168 contactos. Ej. Pentium – Pentium III.
 - **DDR-SDRAM**, 184 contactos. Ej. Athlon XP.



SO-DIMM



SIMM de 72 contactos



DIMM de 168 contactos

Los tres ejemplos ilustran las diferencias entre los productos SIMM, DIMM y SO-DIMM. El DIMM de 168 contactos brinda soporte para transferencias de 64 bits, sin duplicar el tamaño del SIMM de 72 contactos, el cual brinda soporte sólo para transferencias de 32 bits. El SO-DIMM también brinda soporte para transferencias de 32 bits y fue diseñado para su uso en los ordenadores portátiles.

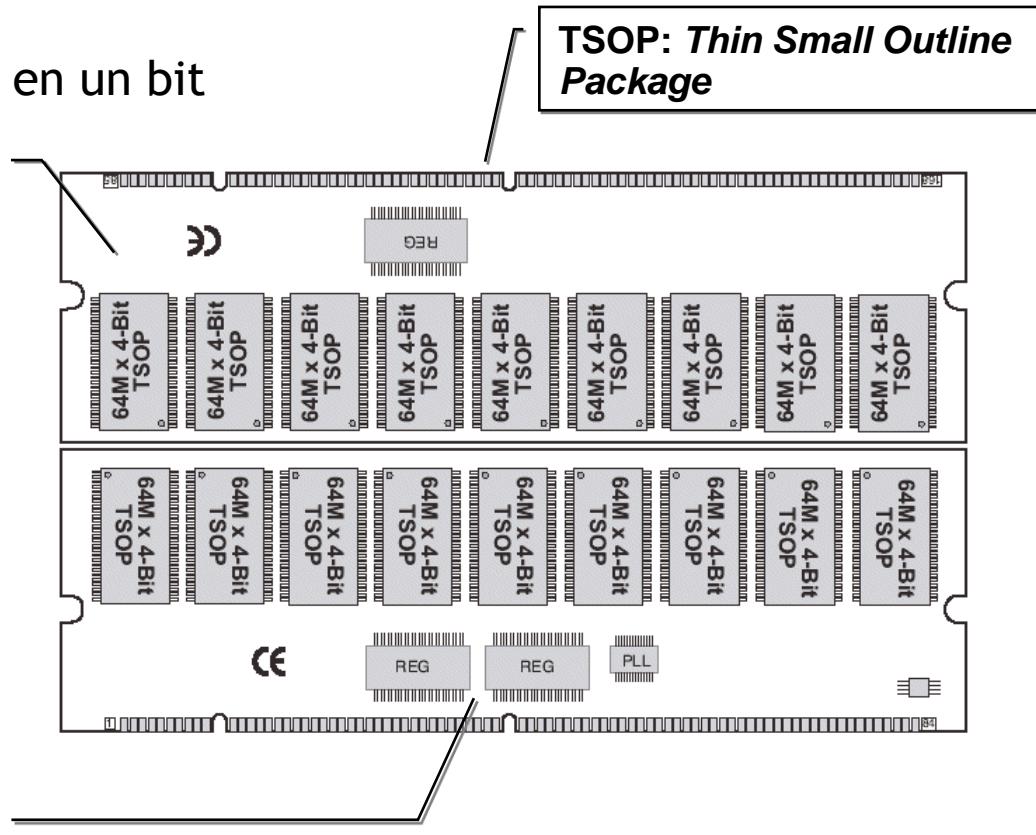
Módulos de memoria RIMM

- Ejemplo: DIMM de 512 MB + ECC ($64M \times (64+8 \text{ (ECC)})$) a 133 MHz.
 - ECC (*Error Checking and Correct / Error-Correcting Code*)
 - Detecta errores en 2, 3 o hasta 4 bits, dependiendo del controlador
 - Corrige error en un bit

18 chips (9 en cada cara)
SDRAM a 133 MHz de $64M \times 4$

Bus de datos: 72 (64 + 8) bits
 $= 18 \text{ chips} \times 4 \text{ bits/chip}$

Registros: amplifican las señales provenientes del chipset en módulos SDRAM grandes



Módulos de memoria SORIMM



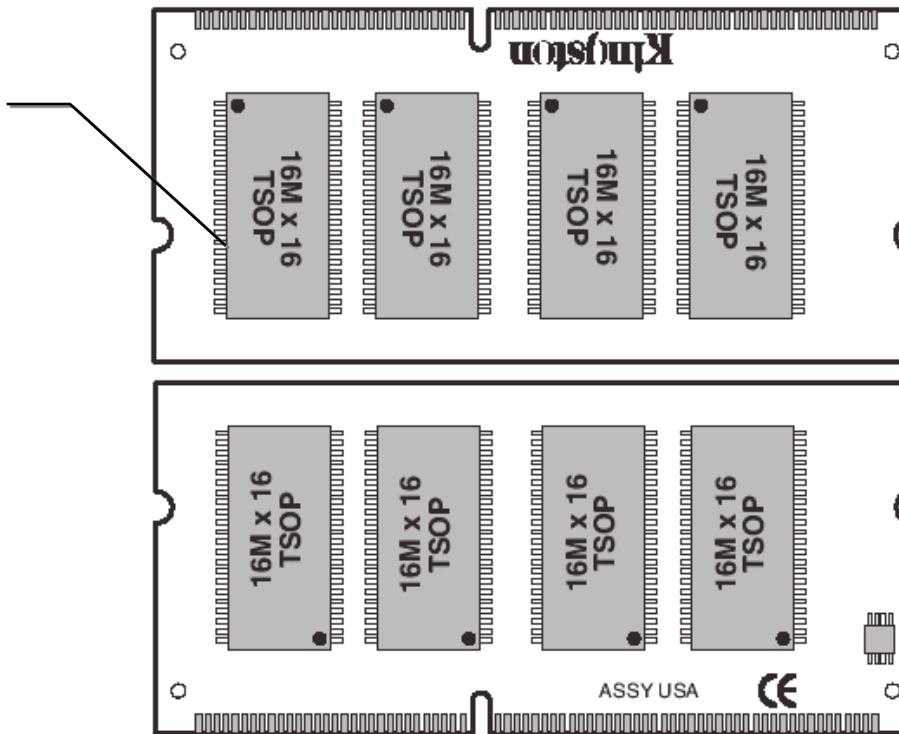
■ SODIMM (*Small Outline DIMM*)

- Mitad de tamaño que un DIMM, 144 contactos.
- Uso en portátiles.
- Ejemplo: SODIMM de 256 MB ($2 \times 16M \times 64$) a 133 MHz:

**8 chips (4 en cada cara)
SDRAM a 133 MHz de
 $16M \times 16$**

**Cada chip tiene 4
bancos de $4M \times 16$**

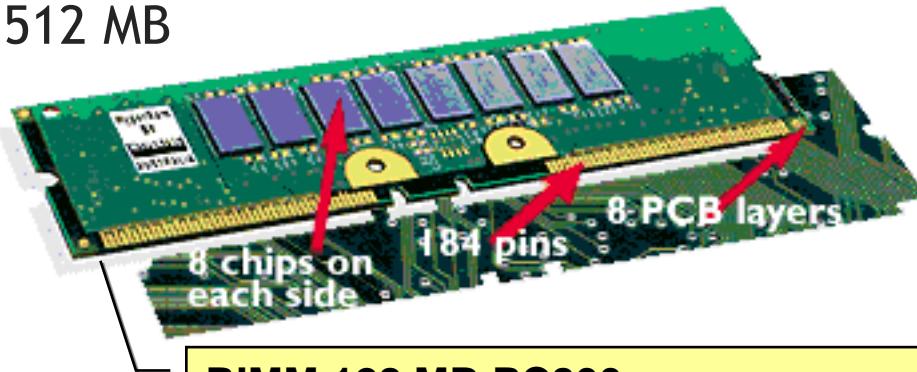
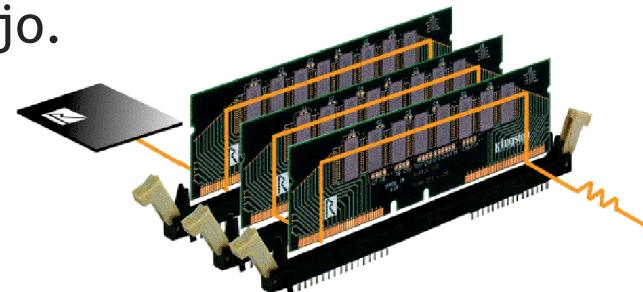
**Bus de datos: 64 bits =
4 chips \times 16 bits/chip
(¡en este caso sí se
amplía el número de
direcciones!)**



Módulos de memoria RIMM

■ **RIMM™ (Rambus In-line Memory Module)**

- Módulos de memoria RDRAM de 184 contactos encontrados en Pentium 4 y estaciones de trabajo.
- Transferencias de 16 bits.
- 1 a 32 chips RDRAM por RIMM.
- Los chips son independientes:
 - Ej.: RIMM de 8 chips y 16 bancos por chip \Rightarrow RIMM tiene 128 bancos.
- 64 MB, 128 MB, 256 MB, 512 MB

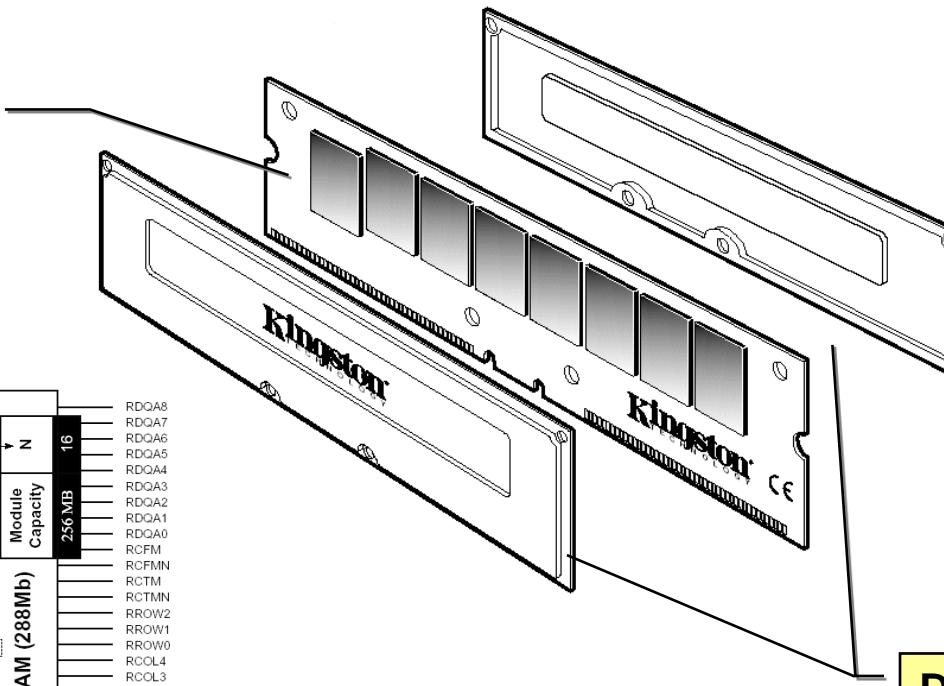
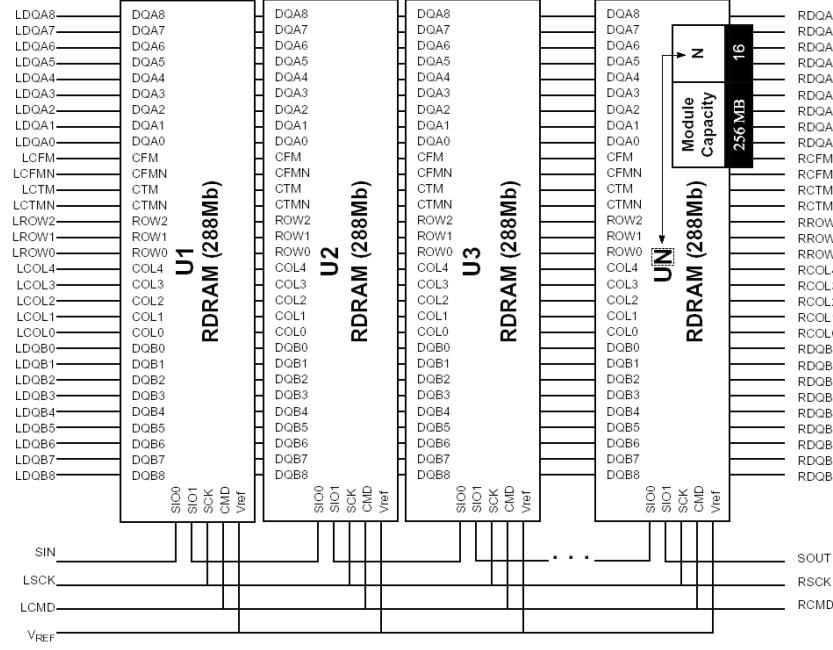


RIMM 128 MB PC800:
16 chips (8 en cada cara) de 4M × 16.

Módulos de memoria RIMM

- Ejemplo: RIMM de 512 MB (256M × 18 con ECC) a 800 Mbps / patilla:

**16 chips (8 en cada cara)
de 288 Mb (16M × 18).**

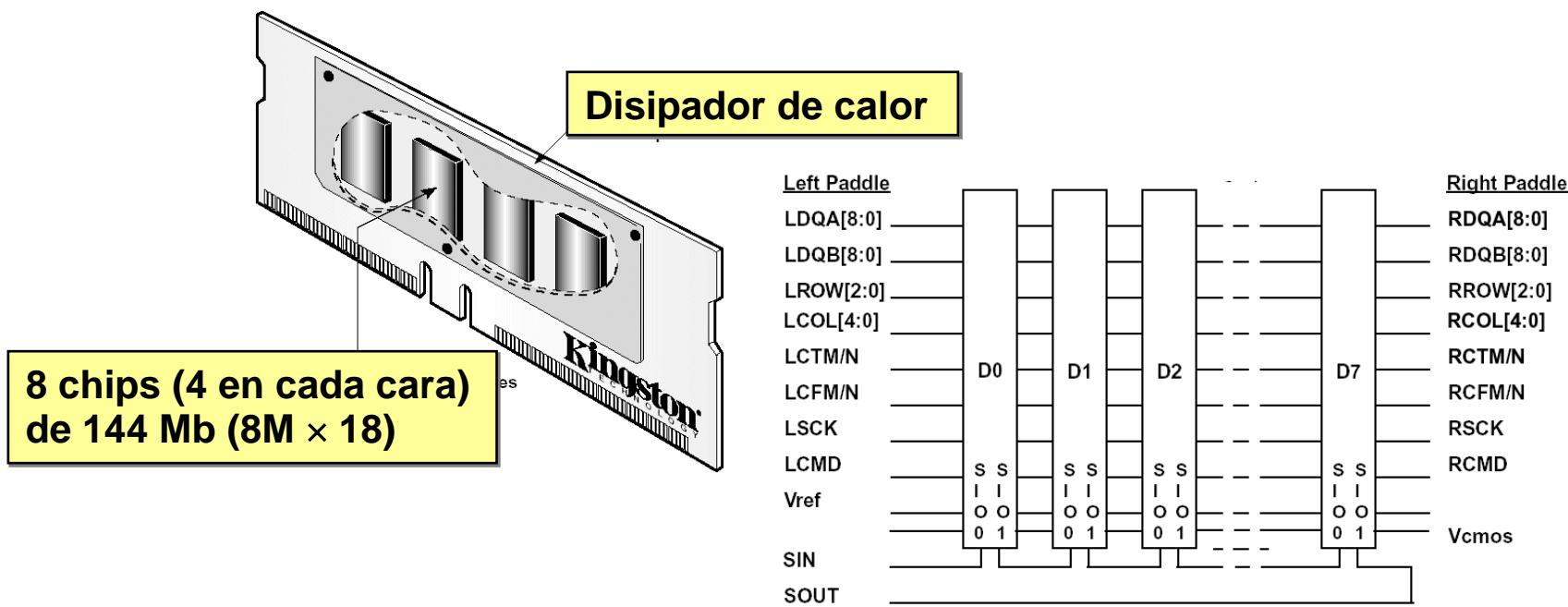


**Disipadores
de calor**

Módulos de memoria SORIMM

■ SORIMM™ (*Small Outline RIMM*)

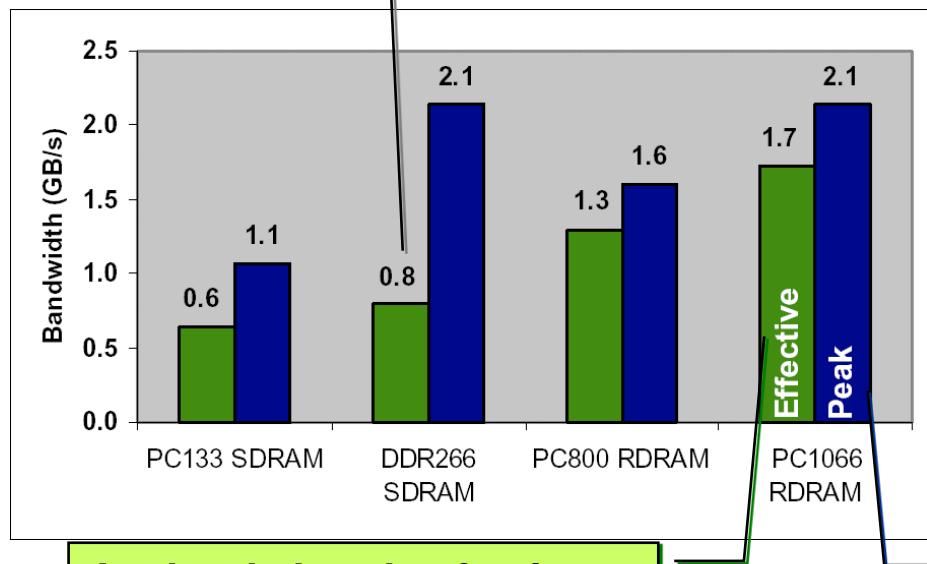
- Módulos RDRAM de 160 contactos para portátiles.
- Ejemplo: SO-RIMM de 128 MB (64M × 18 con ECC) a 800 Mbps / patilla:



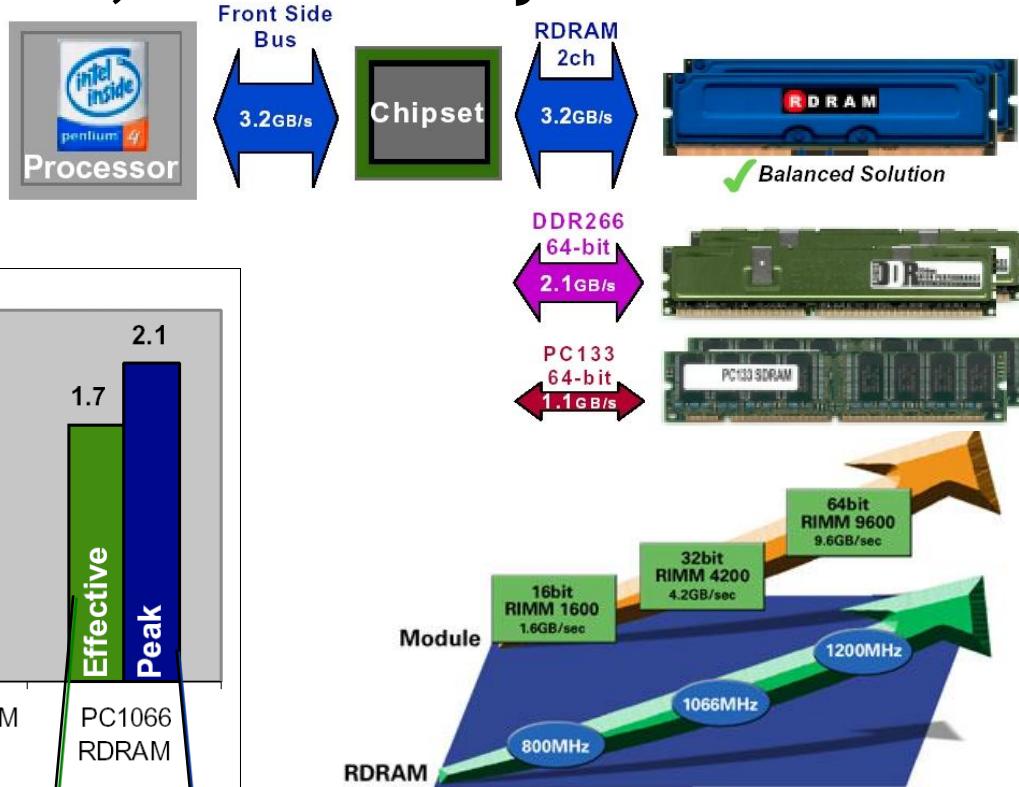
Módulos de memoria en línea

■ Comparación entre SDRAM, DDR SDRAM y RDRAM

¿Por qué el ancho de banda efectivo es menor en la SDRAM?



Ancho de banda efectivo =
Ancho de banda pico ×
eficiencia (% de ancho de
banda pico en un sistema real)

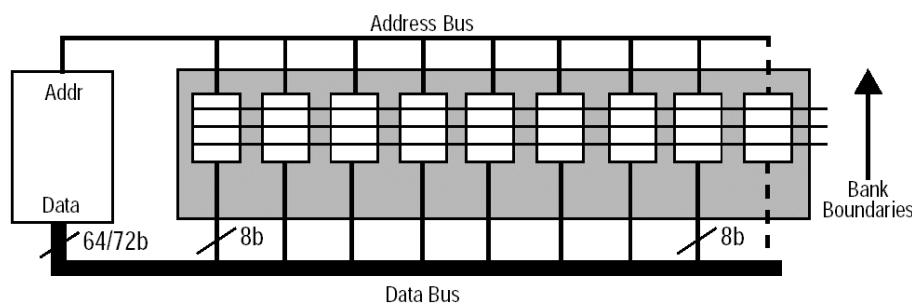


Ancho de banda pico =
Cantidad de datos máxima que
el sistema de memoria puede
proporcionar por unidad de
tiempo bajo supuestos ideales

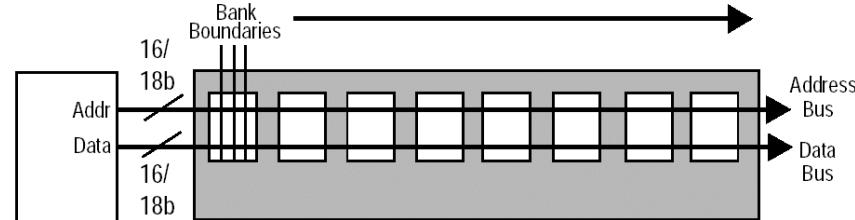
Módulos de memoria en línea

¿Por qué el ancho de banda efectivo es menor en la SDRAM?

SDRAM/DDR SDRAM Module: 8/9 - 32Mx8 Devices, 64b data bus



RDRAM Module: 8 - 256Mb, 4i Devices, Single Channel



- Address Bus loads faster than Data Bus
- 4 banks per device, 4 banks total, banks span across array, 9th device ECC
- **Probability of bank conflict:**

1 outstanding transaction:	1/4	(25%)
2 outstanding transactions:	2/4	(50%)

- Address & Data bus uniformly loaded
- 4 banks per device, 32 banks total, banks additive
- **Probability of bank conflict:**

1 outstanding transaction:	1/32	(3.1%)
2 outstanding transactions:	2/32	(6.3%)

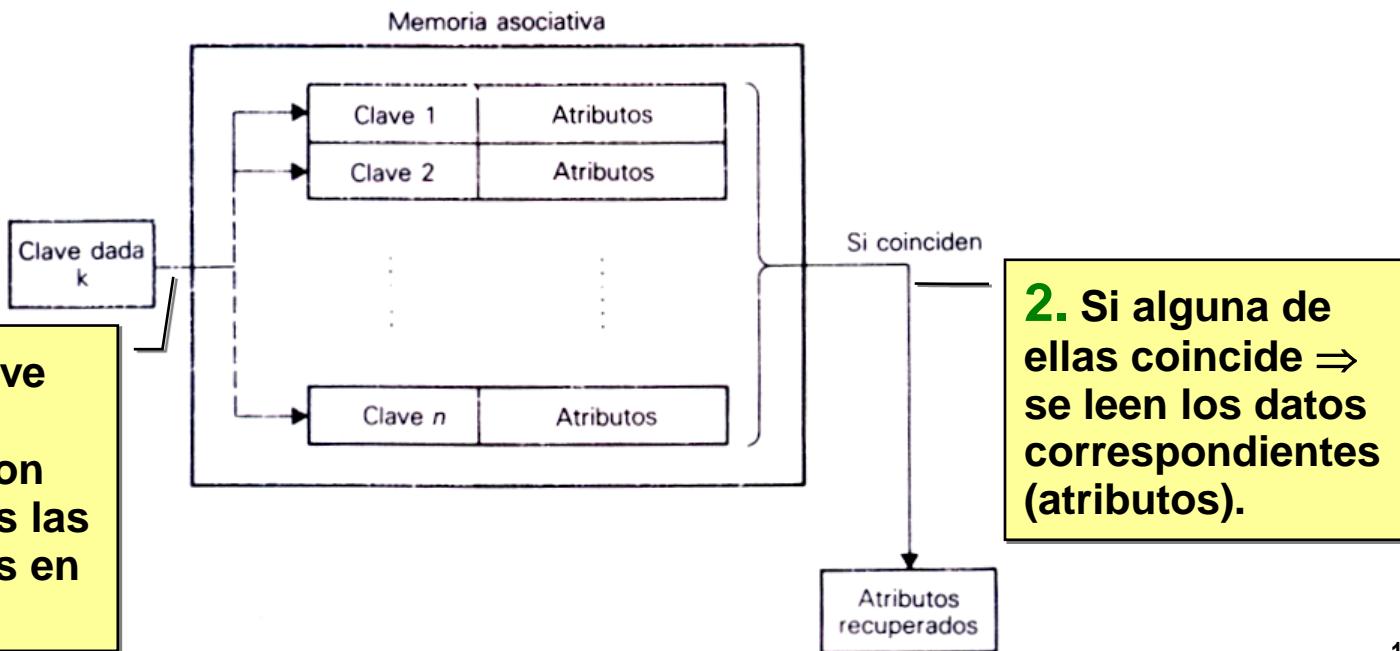
✗ Más conflictos entre bancos al haber menos bancos

Memoria

- **Jerarquía de memoria. Concepto de localidad**
- **Memorias RAM semiconductoras. Memorias de sólo lectura. Prestaciones: velocidad, tamaño y coste**
- **Configuración y diseño de memorias utilizando varios chips**
- **Memorias asociativas**
- **Memoria cache. Influencia en las prestaciones**

Concepto de memoria asociativa

- También llamada:
 - Memoria direccionable por contenido (CAM, *Content Addressable Memory*).
 - Memoria de búsqueda paralela.
 - Memoria multiacceso.
- Es una estructura hardware que permite efectuar operaciones de acceso a los datos (búsqueda, comparación) a gran velocidad.



1. El valor de la clave dada k se compara simultáneamente con los valores de todas las claves almacenadas en memoria.

2. Si alguna de ellas coincide ⇒ se leen los datos correspondientes (atributos).

Concepto de memoria asociativa

✓ El tiempo de búsqueda es muy pequeño ya que el hardware efectúa todas las comparaciones simultáneamente.

La búsqueda se efectúa sólo a partir del contenido de la clave, y no de su dirección.



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



✗ Coste hardware mucho mayor que el de una RAM.

Concepto de memoria asociativa

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 mem. 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 se almacena.
 - El método de acceso es artificial y **complica la programación**.
 - Por ejemplo, para ver quién mide 1,83 hay que realizar una **lenta búsqueda secuencial**.
- Alternativa (memoria asociativa):
 - Emplear uno de los campos de la tabla como clave para acceso al(a los) registro(s) que contiene(n) esa clave.

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.

Concepto de memoria asociativa

■ Operaciones (pensadas para bases de datos):

- Búsquedas extremales:
 - Valor mínimo, máximo, medio.
- Búsquedas de equivalencia:
 - Igual a, no igual a.
 - Similar a (valores idénticos en varios campos).
 - Próximo a (valores que satisfacen cierta condición de proximidad).
- Búsquedas por umbral:
 - Menor que, mayor que.
 - No menor que, no mayor que.
- Búsqueda entre límites.
 - Pertenencia a cierto intervalo (abierto/cerrado).
- Extracciones ordenadas:
 - Ascendentemente, descendentemente.

Concepto de memoria asociativa

■ Aplicaciones:

- Implementación de memorias caché asociativas.
- Implementación de TLB (buffer de traducción anticipada) en memoria virtual.
- Manipulación de información almacenada en bases de datos.
- Enrutadores de red.

Diseño de la memoria asociativa

Estructura de la memoria asociativa:

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

Da las órdenes de comparación, lectura, y escritura.

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

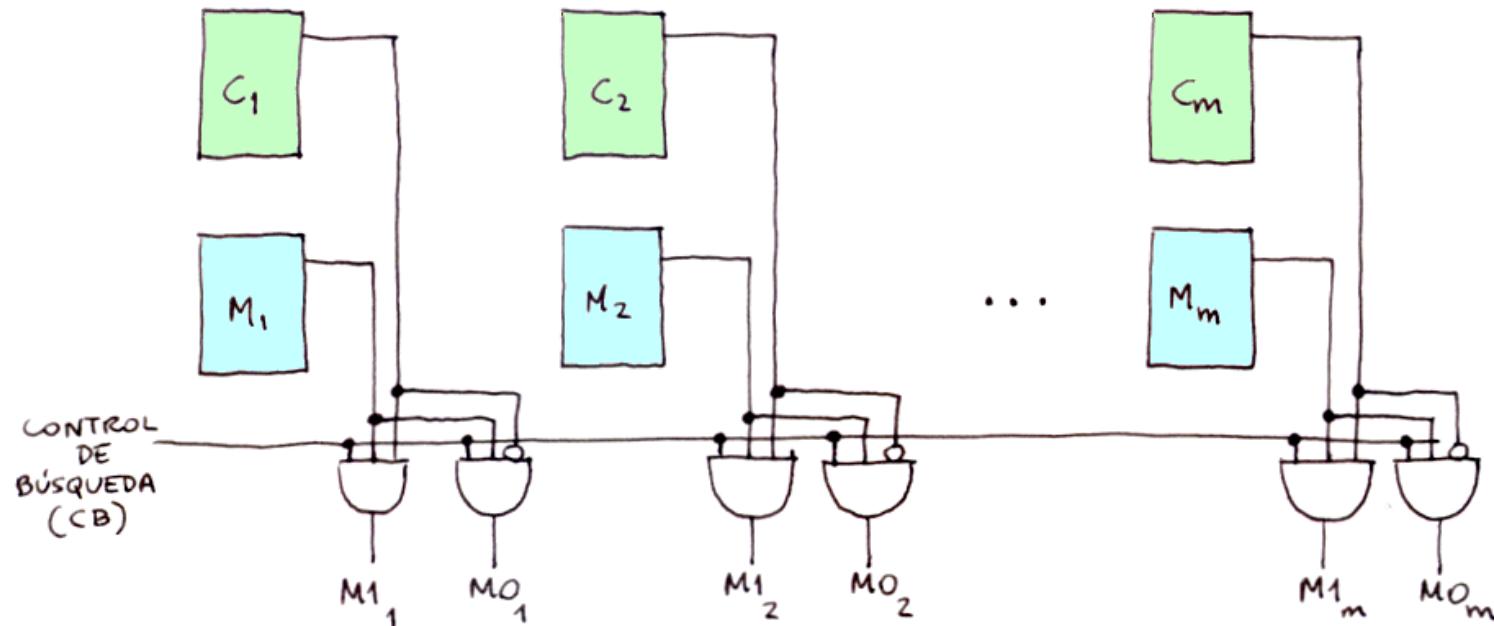
Contiene una de las palabras (o la suma lógica de varias) seleccionada por el proceso de búsqueda.



Colección de celdas básicas de memoria $S_{i,j}$, $1 \leq j \leq m$, cada una con un bit.

Ejemplo de diseño de CAM

■ Lógica de enmascaramiento:



CB	M_j	Líneas de búsqueda	
		$M0_j$	$M1_j$
0	X	0	0
1	0	0	0
1	1	$/C_j$	C_j

Ejemplo de diseño de CAM

■ Lógica de comparación:

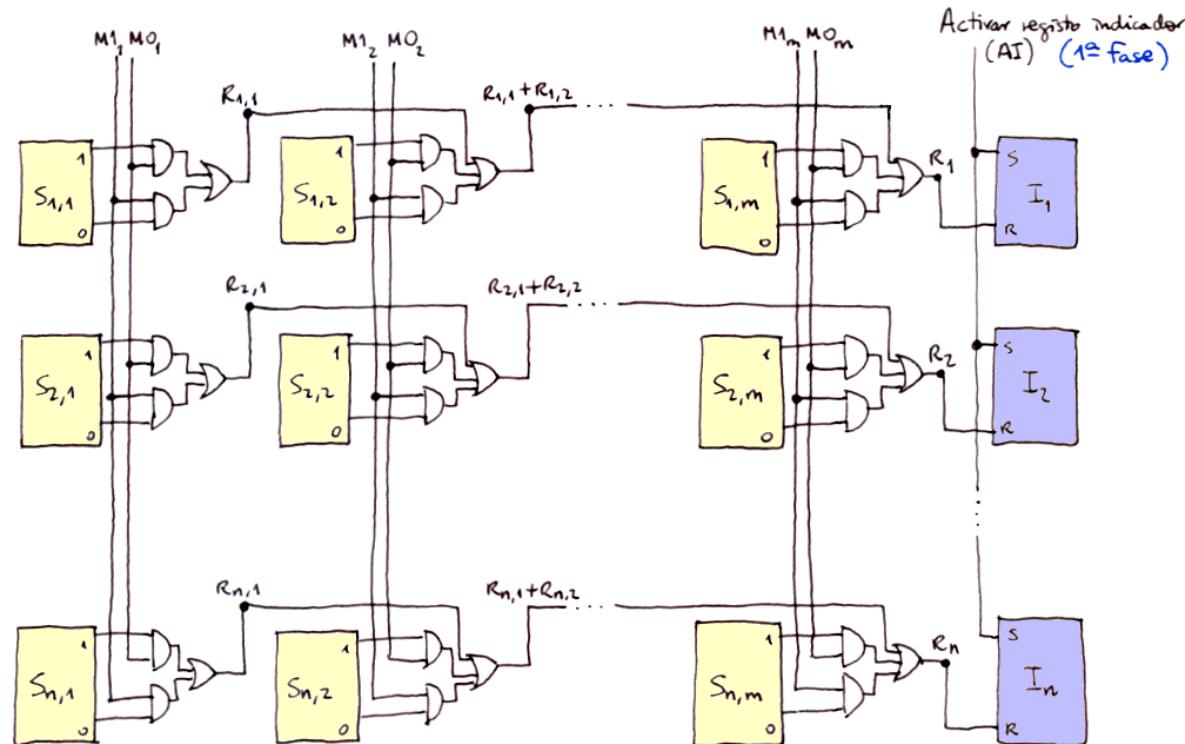
$M_{0,j}$	$M_{1,j}$	$S_{i,j}$	$R_{i,j}$
0	0	X	0
$/C_j$	C_j	C_j	0
$/C_j$	C_j	$/C_j$	1

$$R_i = \sum_{j=1}^m R_{i,j}$$

$$R_{i,j} = 1 \text{ SII } C_j \neq S_{i,j} \text{ y } M_j = 1 \text{ y } CB = 1$$

R_i : señal de desactivación de I_i

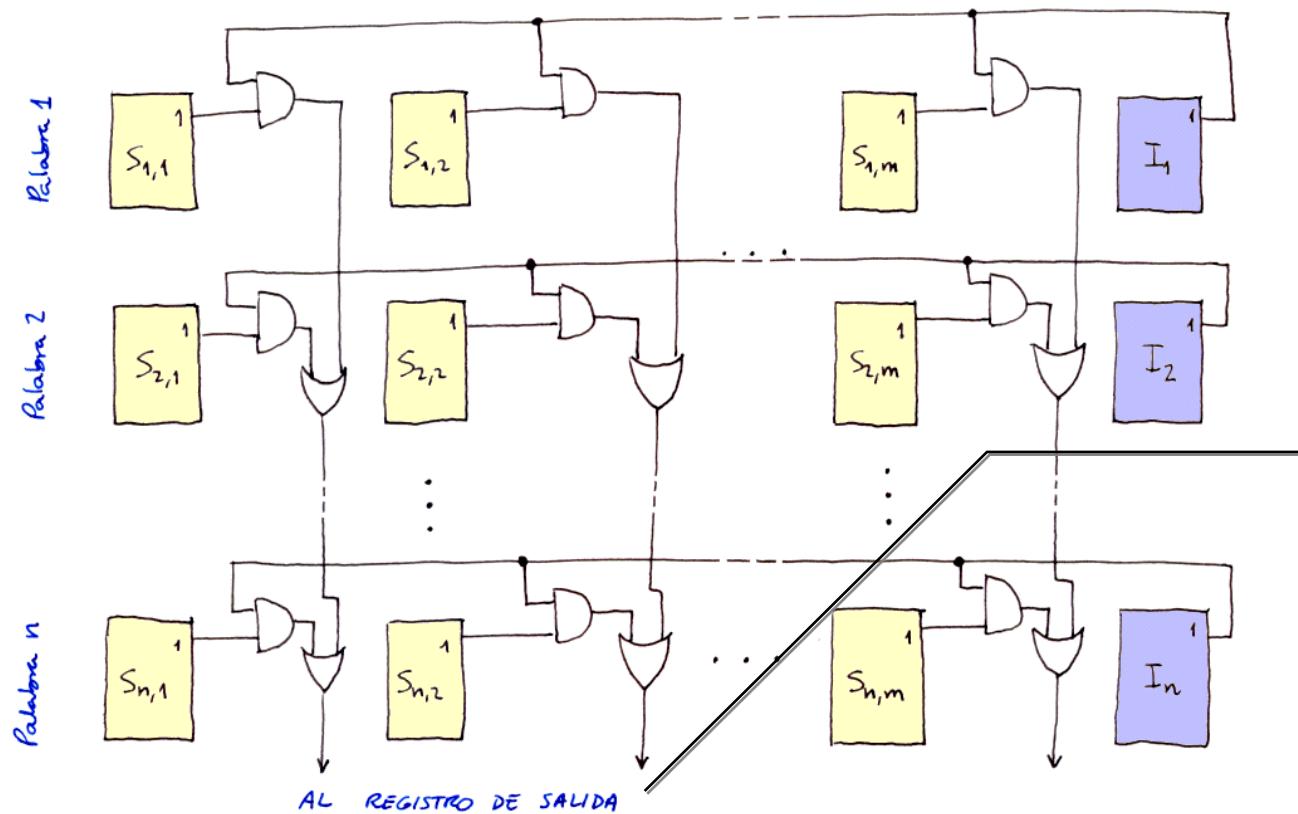
La desactivación del bit I_i ocurre si existe algún bit $S_{i,j}$ de la palabra i que no coincide con C_j . Los bits de C enmascarados no afectan al proceso de comparación.



Ejemplo de diseño de CAM

■ Lectura de una memoria asociativa:

- Dos fases:
 - 1^a: Comparación que deja activos los bits I_i de las celdas a leer.
 - 2^a: Extracción de la información.



El registro de salida contendrá la suma lógica de los contenidos de todas las celdas seleccionadas.

Ejemplo de diseño de CAM

- Si sólo se desea leer la primera palabra:

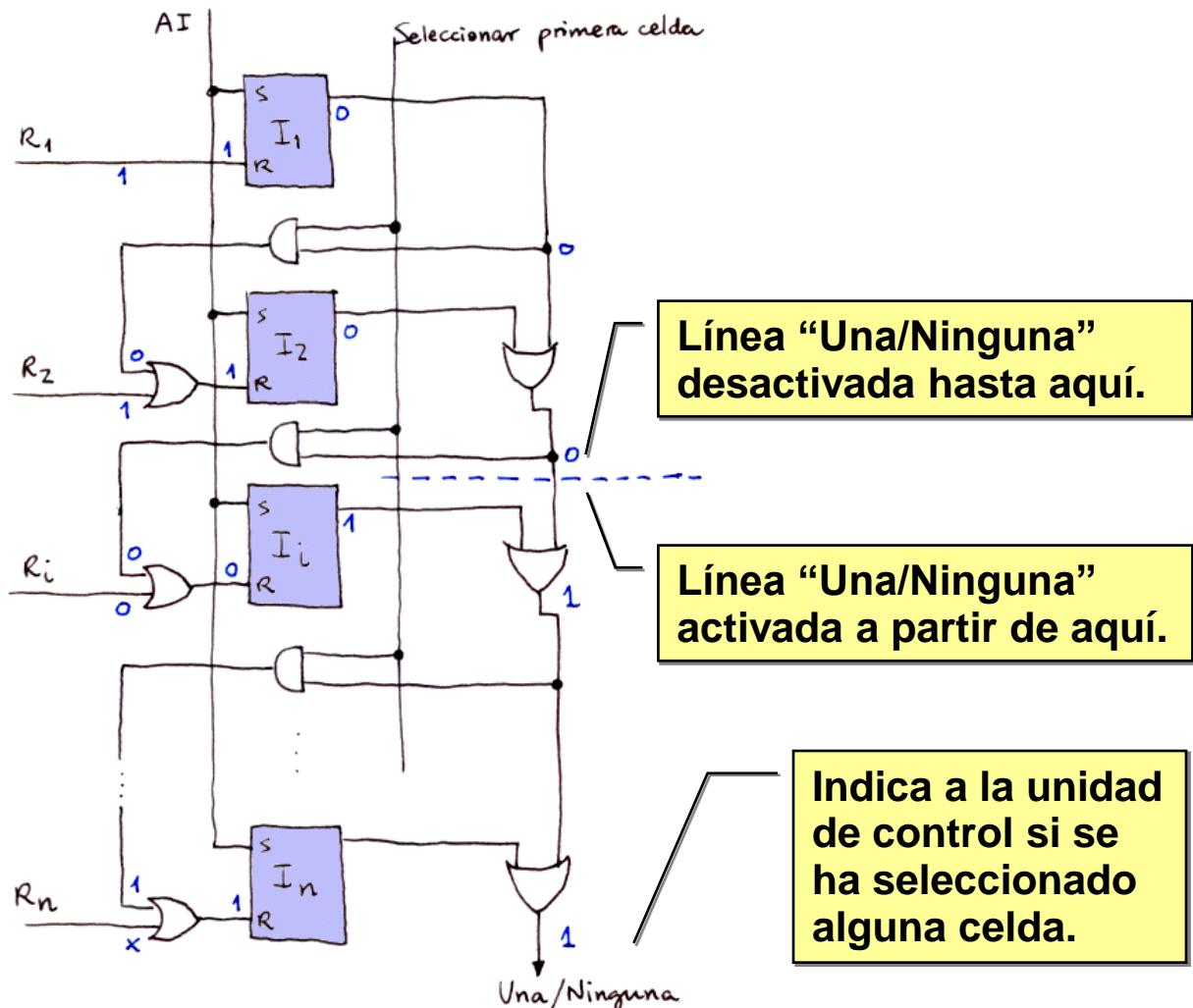
$$I_k=0, k < i$$

$$I_i=1$$

$I_k=0, k>i$ (se desactivan estos bits)

Sólo la palabra i permanece seleccionada y es la que se lee.

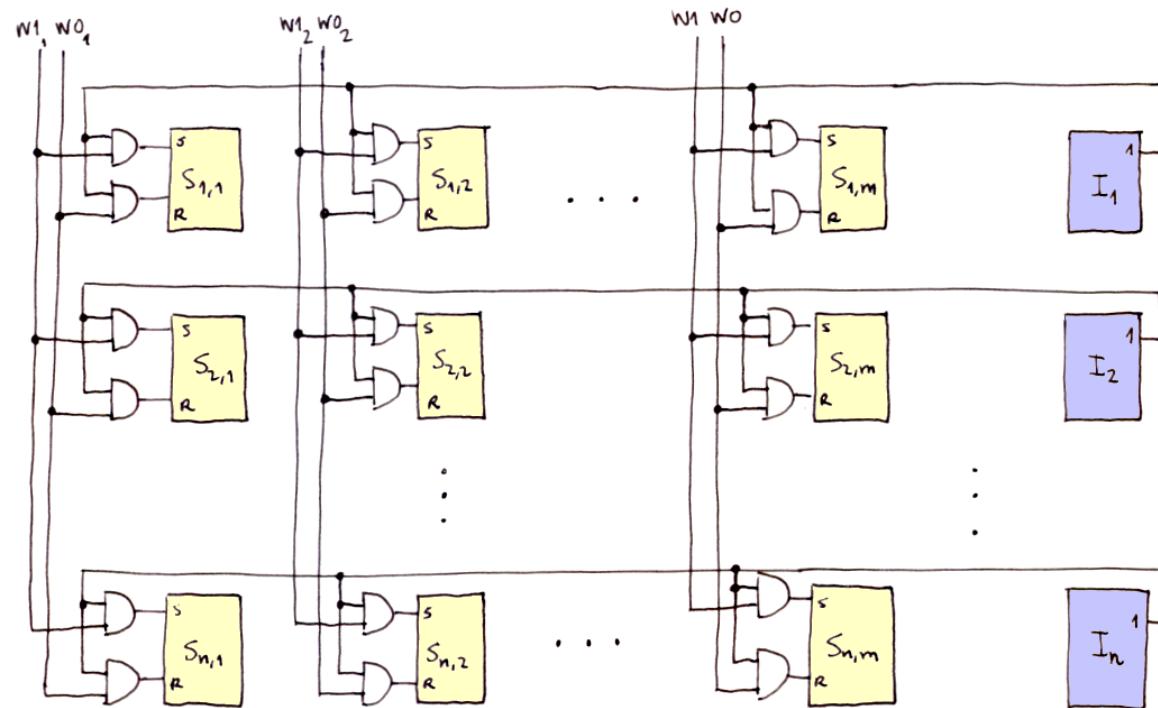
Una vez leída la palabra i -ésima, puede ser marcada de alguna forma. Posterioras comparaciones idénticas permitirán encontrar nuevas palabras.



Ejemplo de diseño de CAM

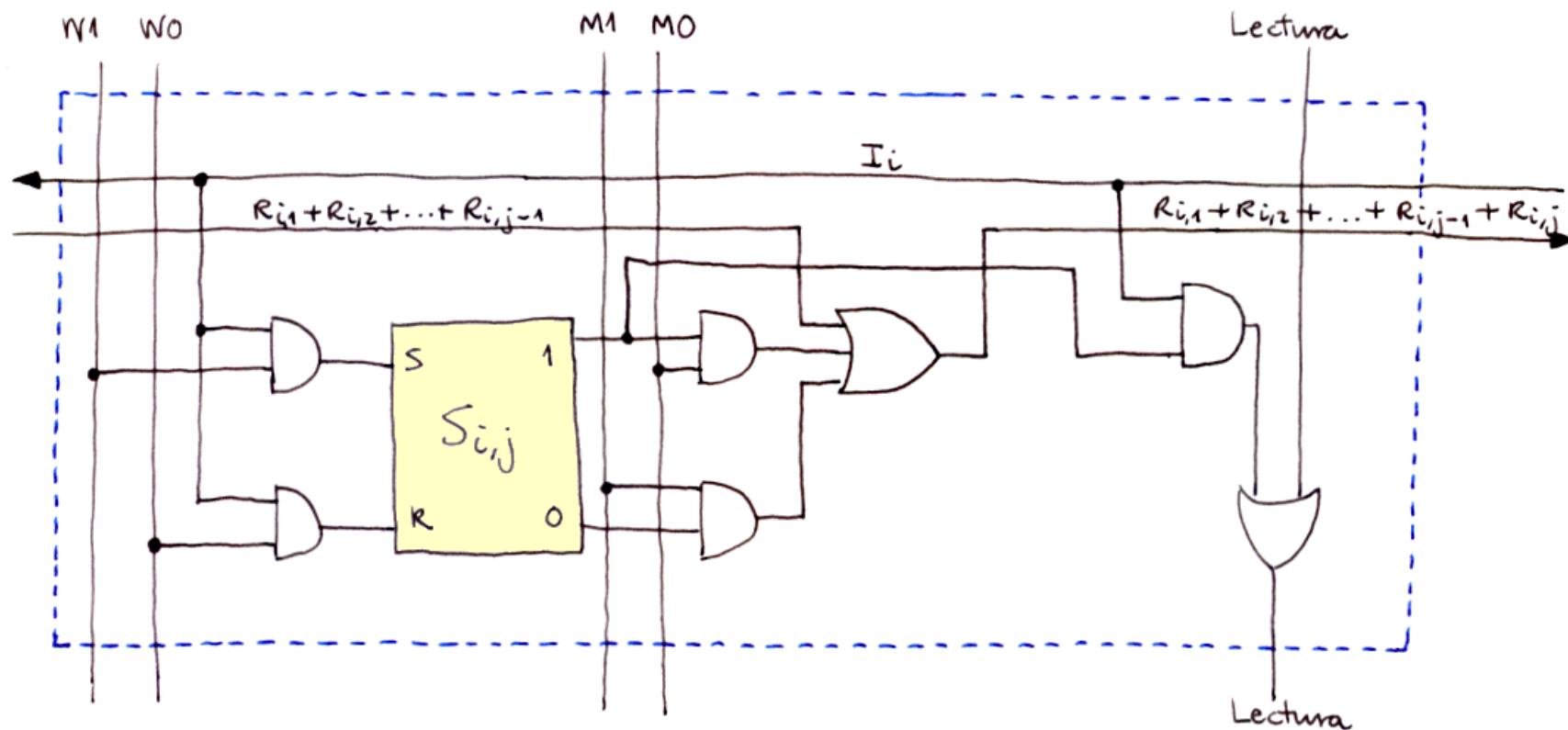
■ Escritura en una memoria asociativa:

- W1, W0 son líneas que llevan la información a escribir.
 - Se generan de forma semejante a las líneas de búsqueda M1, M0 (sustituyendo la señal de control de búsqueda CB por una señal de control de escritura).
- En los bits donde la máscara está desactivada no se escribe.



Ejemplo de diseño de CAM

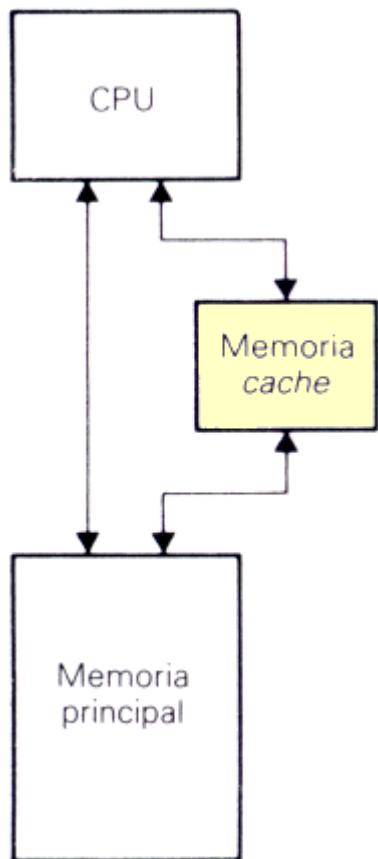
- Estructura global de una celda de un bit:



Memoria

- **Jerarquía de memoria. Concepto de localidad**
- **Memorias RAM semiconductoras. Memorias de sólo lectura. Prestaciones: velocidad, tamaño y coste**
- **Configuración y diseño de memorias utilizando varios chips**
- **Memorias asociativas**
- **Memoria cache. Influencia en las prestaciones**

Caché: concepto



- Memoria pequeña (8KB - 6MB) y rápida (5-10 veces más rápida que MP) situada entre procesador y MP.
 - Niveles más elevados de la jerarquía de memoria.
 - 1^a caché: IBM 360/85, 2^a mitad de los 60.
- En la caché se almacenan aquellas palabras de la MP actualmente en uso.
 - El procesador buscará la información en la caché.
 - Se encuentra en la caché → **acíerto o “cache hit”**
 - Los datos se transfieren de caché al procesador.
 - No se encuentra en la caché → **falta o “cache miss”**
 - Se busca en la MP. Una copia va al procesador, y otra a caché (con su bloque) sustituyéndose un bloque antiguo si no hay sitio.

Caché: concepto

- El éxito de la memoria caché se debe a la propiedad de localidad de los programas.
 - La información que se va a utilizar en un futuro próximo probablemente ya se encuentra en caché.
- Objetivos de diseño en un sistema con 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.

Caché: modelo de evaluación

■ Modelo para evaluar las prestaciones de un sistema que utiliza memoria caché.

- t_c : tiempo de acceso a la memoria caché.
- A : razón de acierto (*hit ratio*) de la memoria caché.
- t_m : tiempo de acceso a la memoria principal.
- Tiempo medio de acceso:

$$\bar{T} = At_c + (1 - A)(t_c + t_m)$$

Acierto \Rightarrow no se accede a MP

Fallo \Rightarrow se accede a caché y a MP

- γ : razón entre los tiempos de acceso a la MP y a la caché

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

Caché: modelo de evaluación

- Eficiencia de un sistema que emplea memoria caché:

$$E = \frac{t_c}{\bar{T}}, \quad 0 < E \leq 1$$

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

- E máxima cuando A=1 (todas las referencias en caché).
- $\gamma \uparrow \Rightarrow E \downarrow$; $A \downarrow \Rightarrow E \downarrow$. Interesa γ baja y A alta.
- Ejemplo:

$$\bullet \quad t_c = 15 \text{ ns} \quad \bar{T} = At_c + (1-A)(t_c + t_m) = 0,9 \cdot 15 + 0,1 \cdot 115 = 25 \text{ ns}$$

$$\bullet \quad t_m = 100 \text{ ns} \quad \gamma = \frac{t_m}{t_c} = \frac{100}{15} = 6,6 \quad E = \frac{1}{1 + \gamma(1-A)} = \frac{1}{1 + 6,6 \cdot 0,1} = 0,6$$

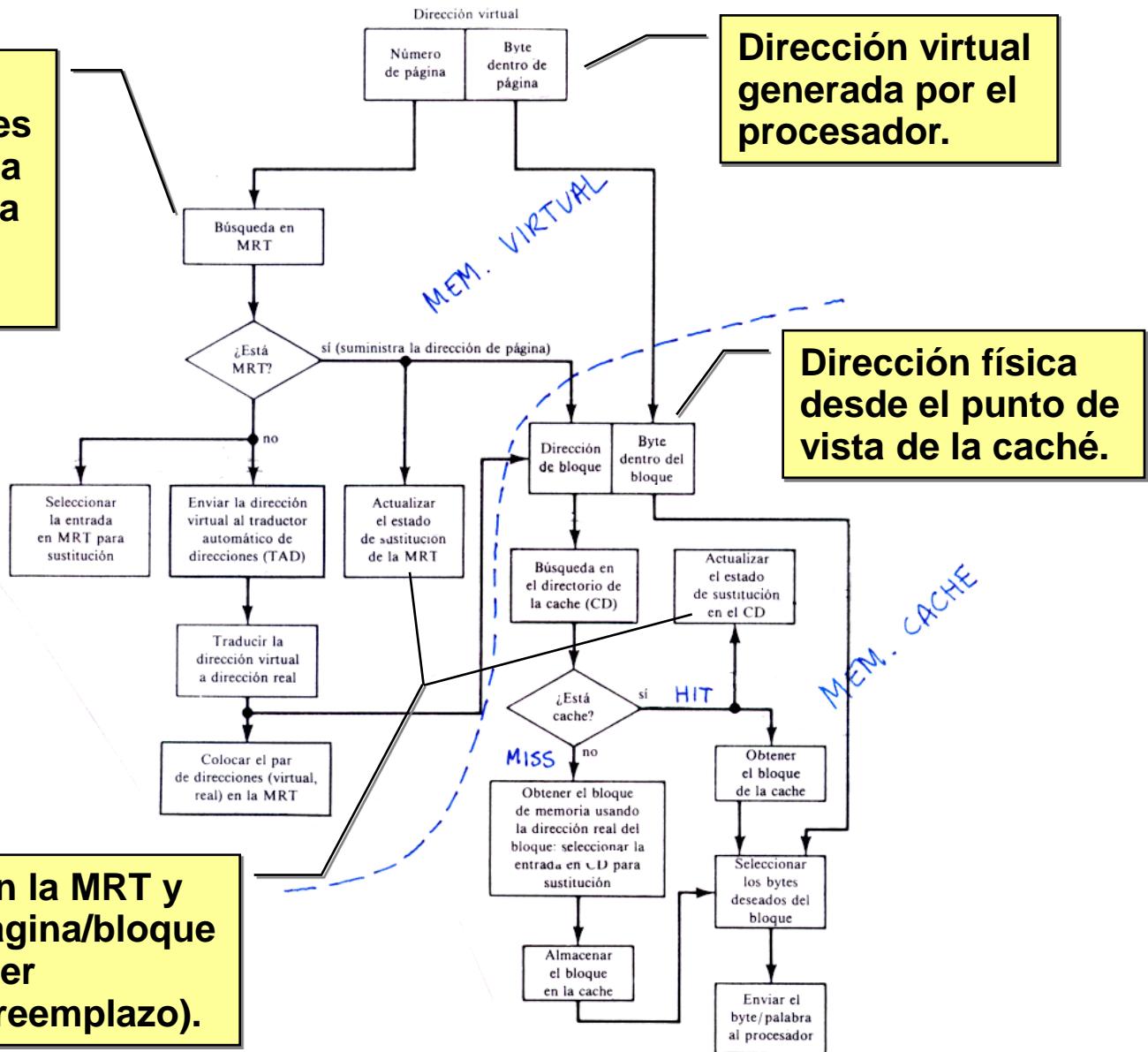
Caché: estructura y funcionamiento

- Zona de almacenamiento.
 - Memoria **SRAM**.
 - Particionada en un **conjunto de bloques** o líneas.
 - **Bloque o línea**: unidad básica de información que se puede transferir entre la caché y la MP.
 - Cada bloque de MP va a un **marco de bloque** en caché.
 - La MP tiene un gran número de bloques; 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 en principio.
- Directorio caché.
 - Se implementa mediante memoria SRAM y comparadores o mediante una memoria asociativa.
 - Guarda las **direcciones de los bloques (etiquetas o 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:
 - **Colección de pares dirección-bloque** (dirección del bloque en MP-copia del contenido del bloque).

Caché: estructura y funcionamiento

MRT: Memoria de Reserva de Traducciones
(TLB): pequeña memoria asociativa que almacena pares dirección virtual-dirección física.

Dirección virtual generada por el procesador.



Estados de sustitución en la MRT y el CD: determinan qué página/bloque respectivamente deben ser sustituidos (políticas de reemplazo).

Caché: políticas y otros parámetros

- **Parámetros determinantes en el correcto funcionamiento y efectividad de un sistema de caché:**
 - Política de **extracción** o algoritmo de búsqueda de 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 MP?
 - Política o algoritmo de **reemplazo**.
 - Cuando se produce una falta y la caché está llena, ¿qué bloque se sustituye por uno nuevo?
 - Política o algoritmo de **actualización** de MP.
 - ¿Cuándo debe modificarse el contenido de MP tras escribir en caché?
 - Cachés separadas frente a caché unificada.
 - Tamaños de bloque y de caché.
 - Diseño del sistema de memoria para dar soporte a cachés.

Caché: política de extracción

■ Política de extracción o algoritmo de búsqueda

- Decide cuándo y qué información se va a introducir en 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 necesaria.
- Políticas de extracción selectivas.
 - Por demanda o anticipativas si se establecen diferencias entre datos que pueden pasar a la caché y datos que no.

Caché: política de extracción

■ Búsqueda por demanda:

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

■ Prebúsqueda o preextracción (políticas anticipativas):

- Algoritmo más complejo y eficiente.
- Busca el bloque que el procesador va a necesitar antes de que lo solicite, reduciendo la probabilidad de falta.
- Localidad espacial \Rightarrow se preextrae usualmente el bloque siguiente al solicitado por el procesador.

...

Caché: política de extracción

- Cuándo realizar la preextracción:
 - **Preextracción siempre:**
 - Se realiza la prebúsqueda del bloque $i+1$ cada vez que el procesador referencia el bloque i . Incrementa el tráfico entre caché y MP.
 - **Preextracción por falta:**
 - 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 referencie por primera vez a un bloque i previamente preextraido.
- Problema:
 - se introducen en la caché bloques que posiblemente el procesador nunca referencie. Si la caché es pequeña, estos bloques reemplazan a otros que sí pueden necesitarse.

Caché: política de colocación

■ 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 MP.
 - Marcos de bloque o líneas en la caché.

La política de colocación determina la función de correspondencia entre bloques de la MP y marcos de bloque de la caché.

- Puesto que la MP es mucho mayor que la caché, cada marco de bloque almacenará a lo largo del tiempo muchos bloques de la MP.

Caché: política de colocación

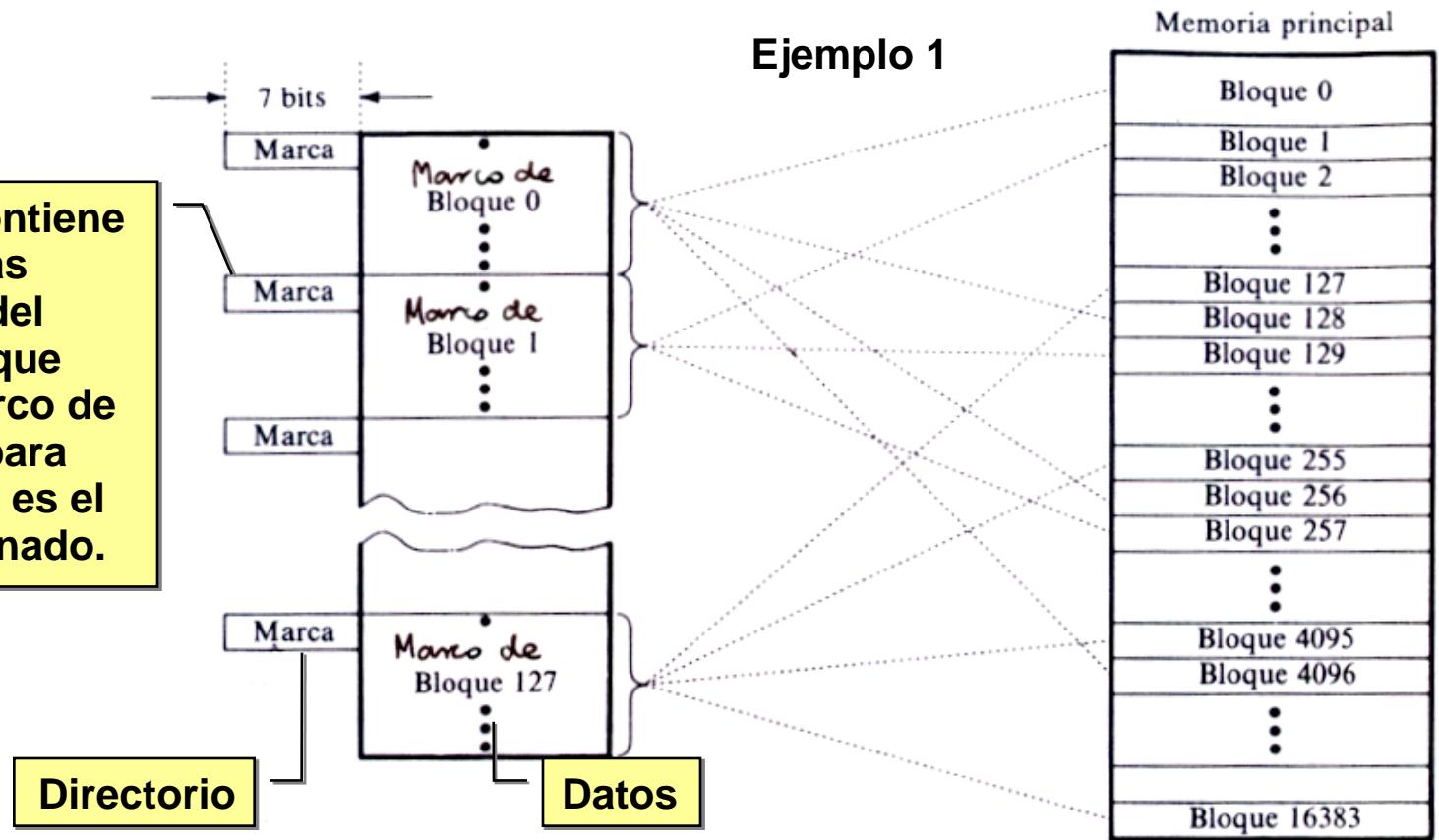
- Existen cuatro políticas de colocación:
 - Correspondencia directa.
 - Totalmente asociativa.
 - Asociativa por conjuntos.
 - Correspondencia por sectores.
 - Consideraremos dos ejemplos:
 - Ejemplo 1:
 - Tamaño de caché: 2K palabras.
 - 16 palabras por bloque.
 - Memoria principal: 256K palabras.
 - Dirección física: 18 bits.
 - Ejemplo 2:
 - Tamaño de caché: 2^{2+w} palabras.
 - 2^w palabras por bloque.
 - Memoria principal: 2^{4+w} palabras.
 - Dirección física: $4+w$ bits.
 - La MP tiene 2^N bloques, la caché tiene 2^n marcos de bloque y un bloque tiene 2^w palabras. Una dirección física tendrá $N+w$ bits.
 - Ejemplo 1: $N=14$, $n=7$, $w=4$
 - Ejemplo 2: $N=4$, $n=2$
- $\left. \begin{matrix} \\ \\ \end{matrix} \right\} \Rightarrow 128 \text{ marcos de bloque}$ $\left. \begin{matrix} \\ \\ \end{matrix} \right\} \Rightarrow 16K \text{ bloques en MP}$
 $\left. \begin{matrix} \\ \\ \end{matrix} \right\} \Rightarrow 4 \text{ marcos de bloque}$ $\left. \begin{matrix} \\ \\ \end{matrix} \right\} \Rightarrow 16 \text{ bloques en MP}$

Caché: política de colocación

■ Correspondencia directa

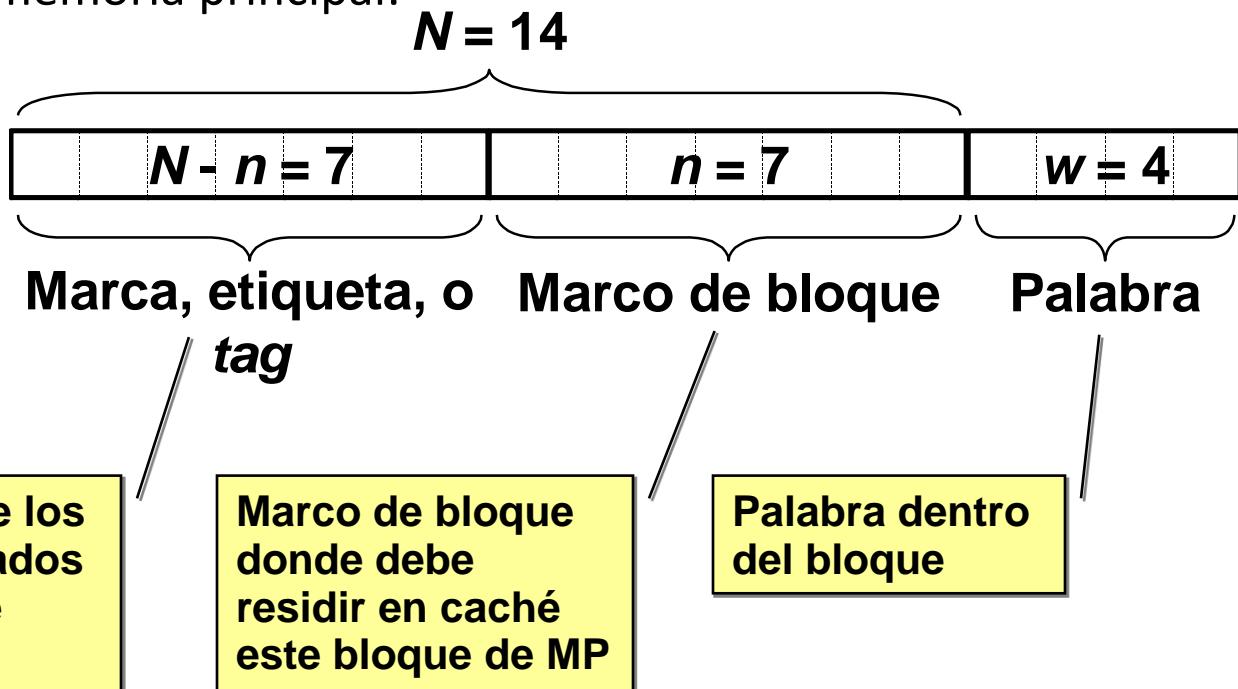
- Bloque i de MP \Rightarrow marco de bloque $i \bmod 2^n$ de caché.
- A cada marco de bloque le corresponde sólo un subconjunto de bloques de MP.

Cada marca contiene los $N-n$ bits más significativos del bloque de MP que hay en ese marco de bloque. Sirve para identificar cuál es el bloque almacenado.



Caché: política de colocación

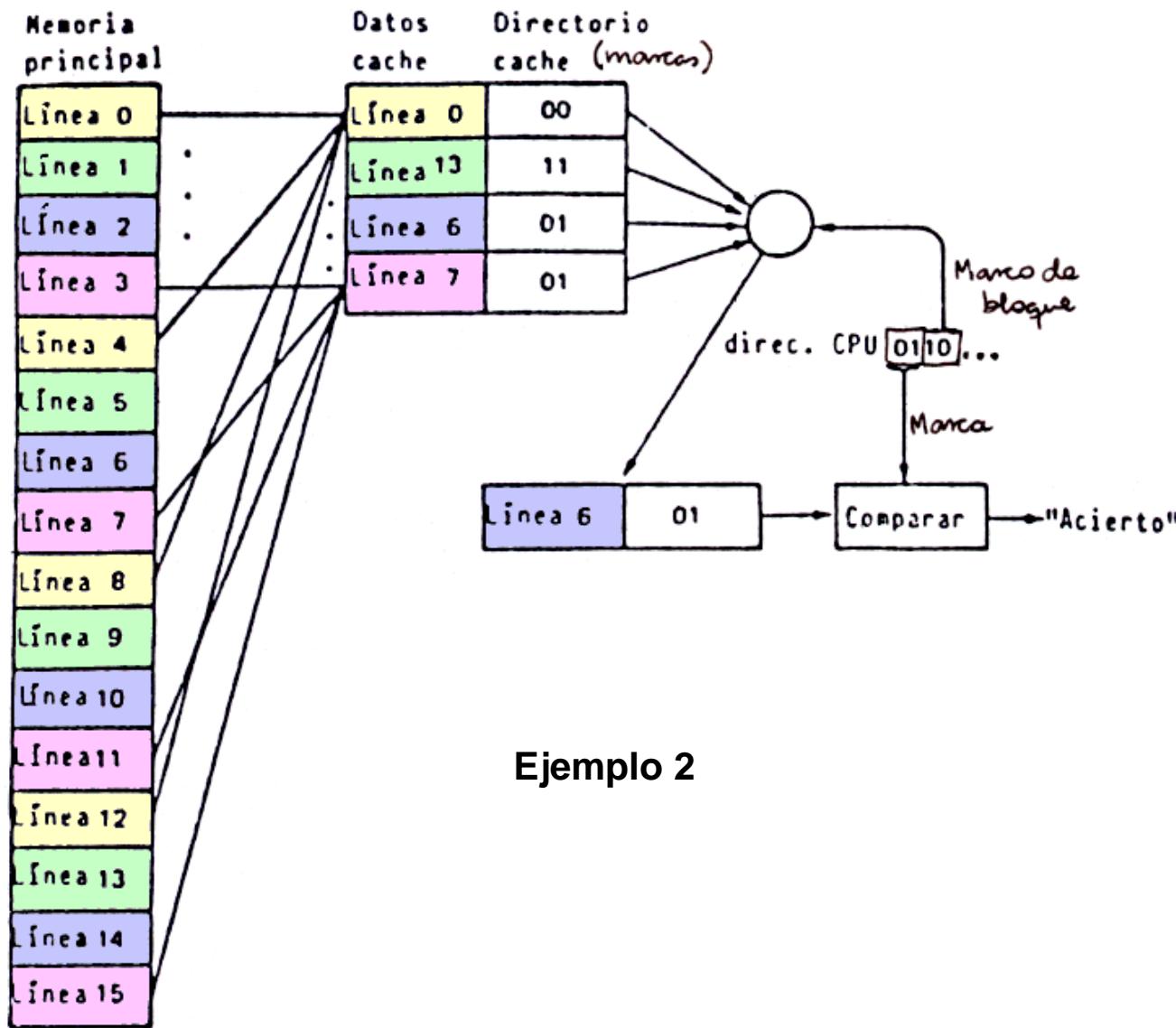
- Dirección de memoria principal:



✓ Simplicidad y bajo coste.

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

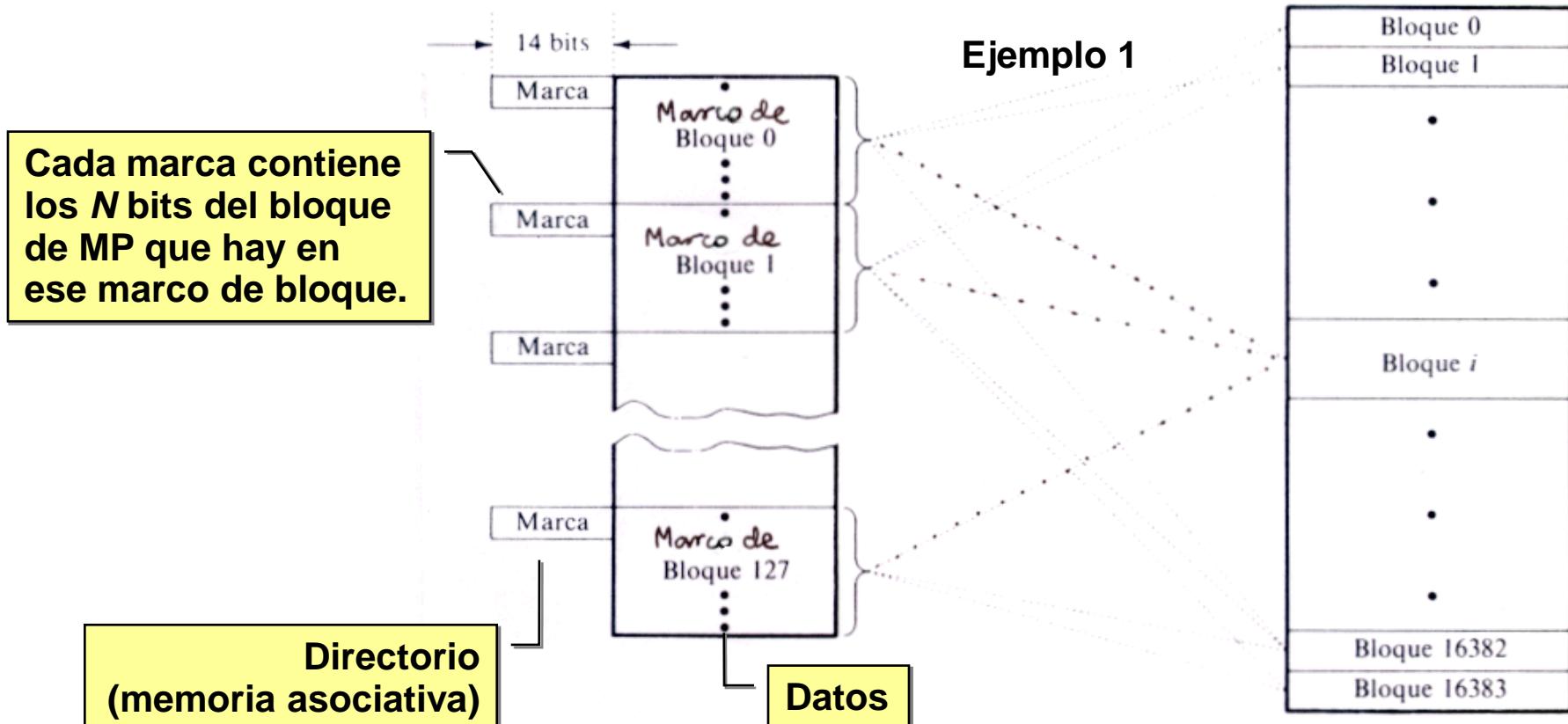
Caché: política de colocación



Caché: política de colocación

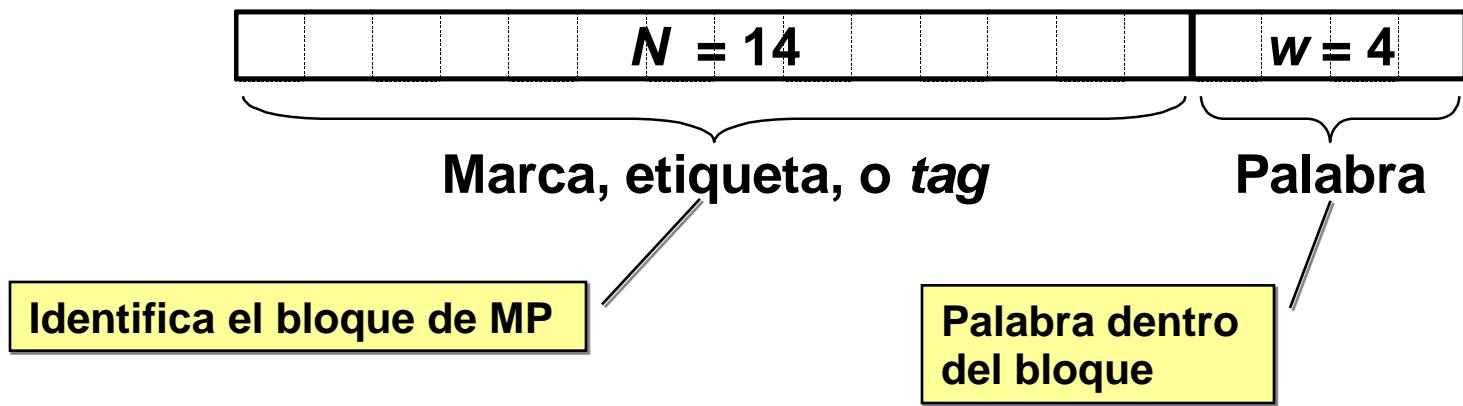
■ Correspondencia totalmente asociativa

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



Caché: política de colocación

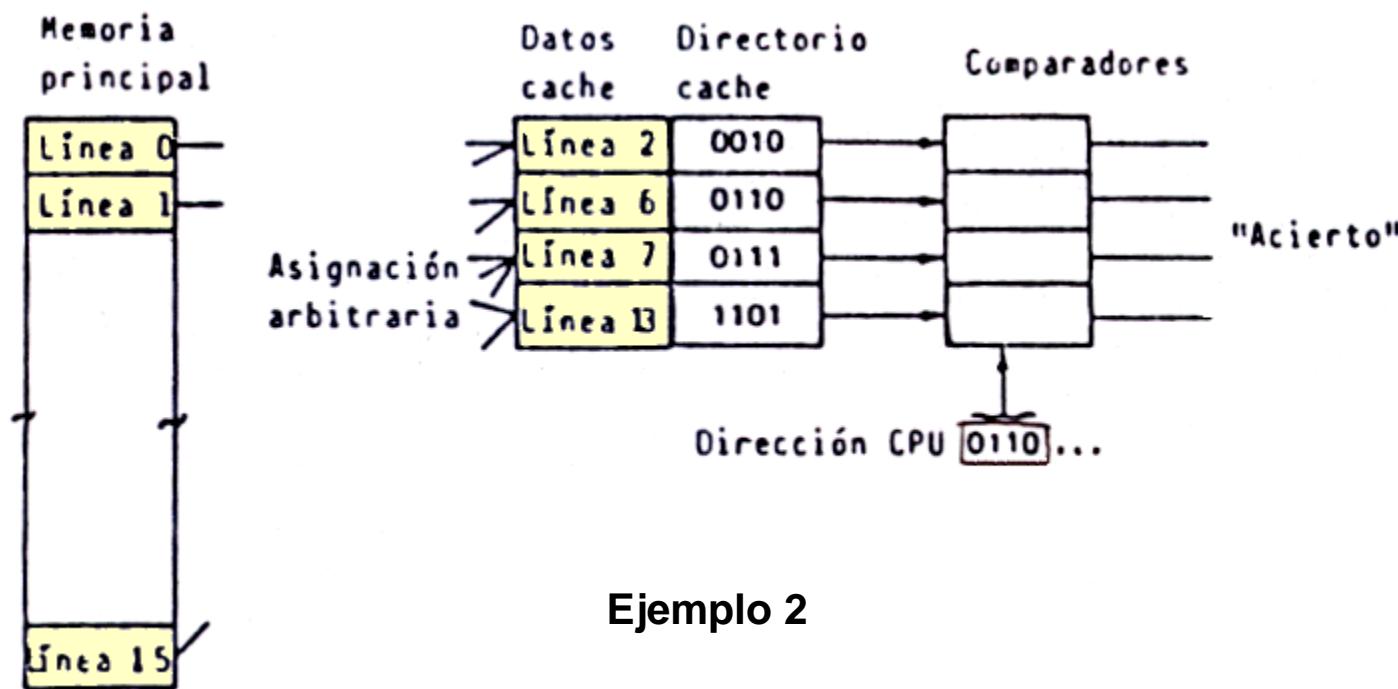
- Dirección de memoria principal:



✓ **Flexible.** Permite cualquier combinación de bloques de MP en la caché. Elimina en gran medida conflictos entre bloques.

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

Caché: política de colocación



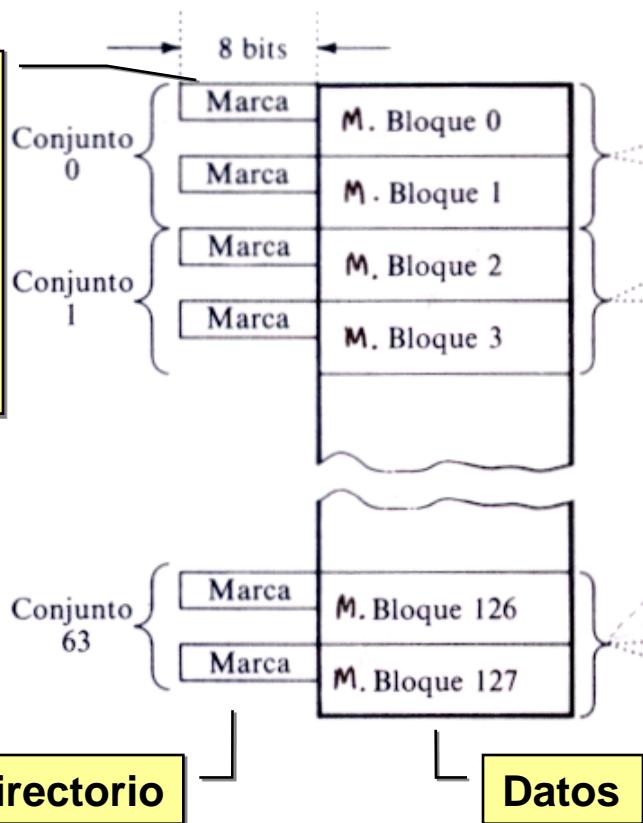
Caché: política de colocación

■ Correspondencia asociativa por conjuntos

- La caché se subdivide en 2^c conjuntos disjuntos
 - 2^{n-c} marcos de bloque / conjunto
- Bloque i de MP \Rightarrow conjunto $i \bmod 2^c$ de caché. Dentro de ese conjunto puede estar en cualquier marco de bloque.
- Hay dos fases en el acceso a caché:
 - Selección directa del conjunto donde puede estar ese bloque.
 - Búsqueda asociativa (dentro del conjunto) de la marca.
- A la correspondencia asociativa por conjuntos con 2^{n-c} marcos de bloque / conjunto también se le llama correspondencia asociativa de 2^{n-c} vías.
 - La vía i está formada por todos los marcos de bloque de la caché que ocupan el lugar i -ésimo dentro de su conjunto.
 - Tanto el ejemplo 1 como el 2 corresponden a caches asociativas de 2 vías.

Caché: política de colocación

Cada marca contiene los N_c bits más significativos del bloque de MP que hay en ese marco de bloque.

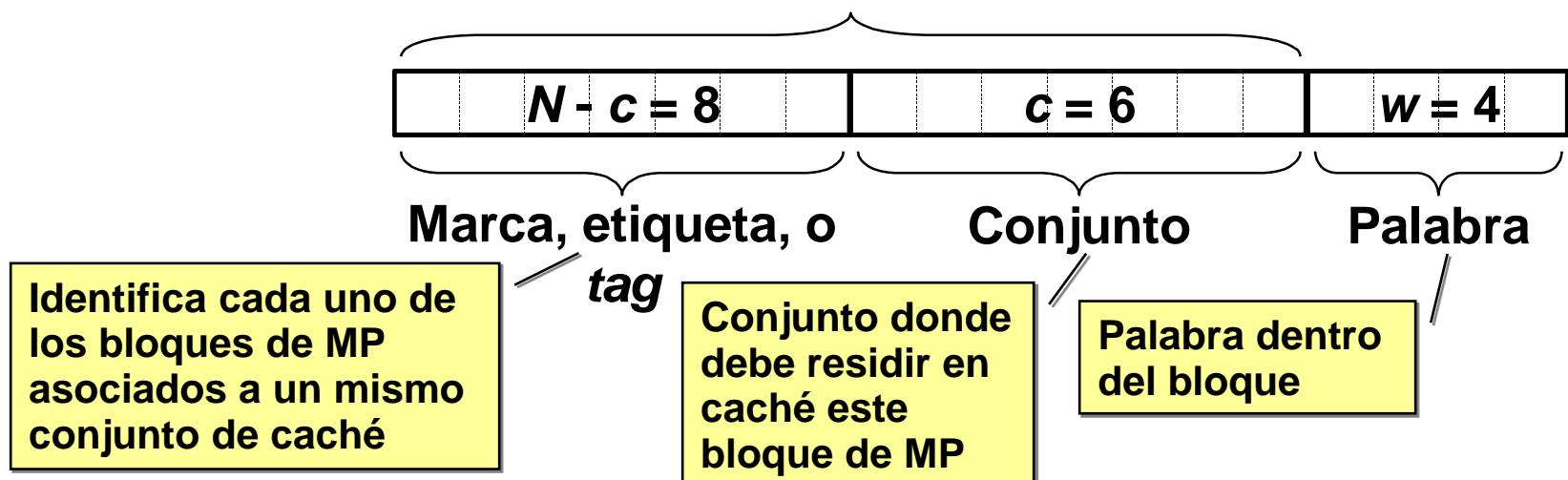


Ejemplo 1

Bloque 0
Bloque 1
⋮
Bloque 63
Bloque 64
Bloque 65
⋮
Bloque 4095
Bloque 16383

Caché: política de colocación

- Dirección de memoria principal: $N = 14$



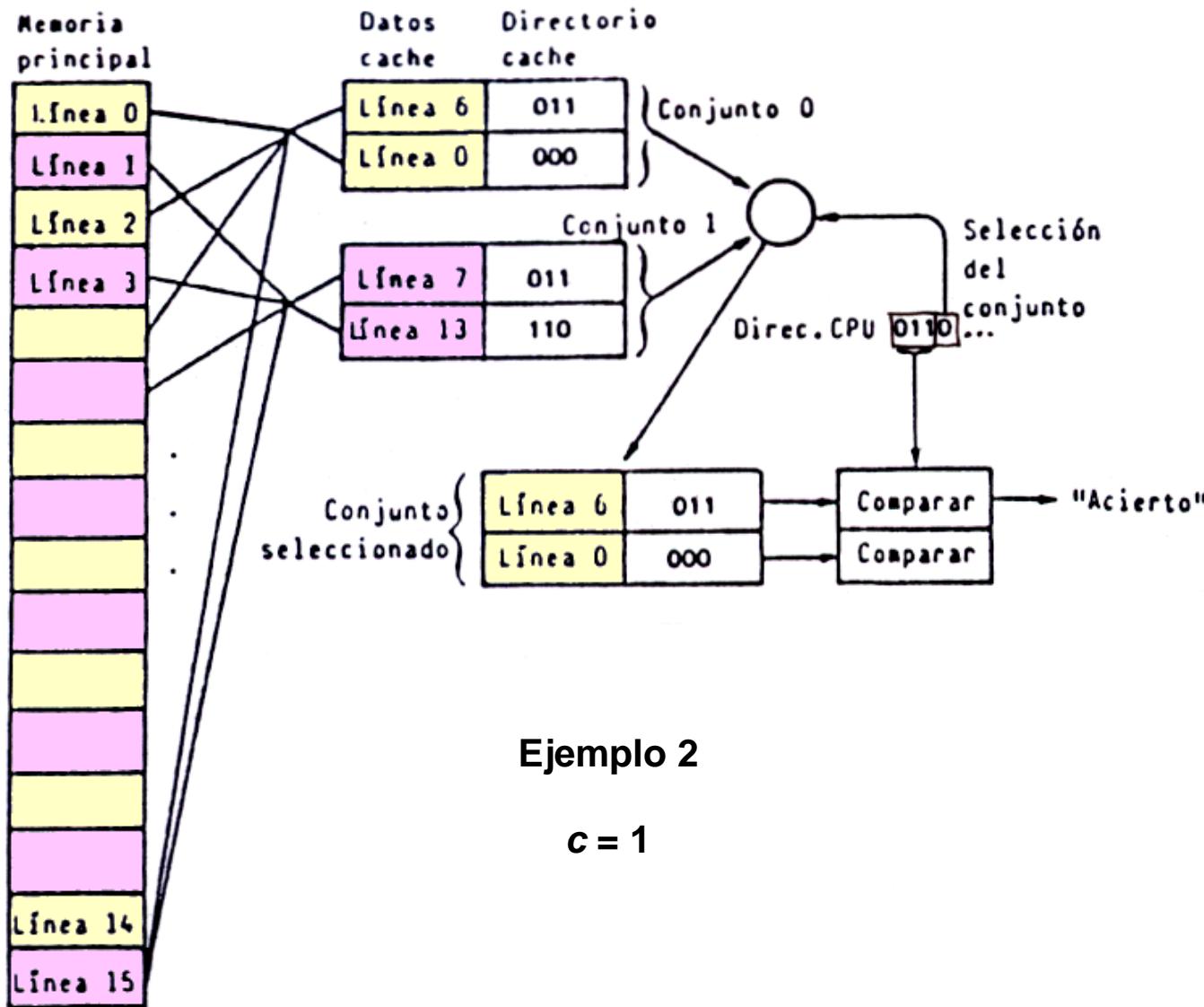
$c = 0$ (1 conjunto) \Rightarrow corresp. totalmente asociativa.

$c = n$ (1 marco de bloque / conjunto) \Rightarrow corresp. directa.

$0 < c < n \Rightarrow$ se pretende reducir el **coste de la totalmente asociativa** manteniendo un **rendimiento similar** \Rightarrow es la técnica más utilizada.

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

Caché: política de colocación



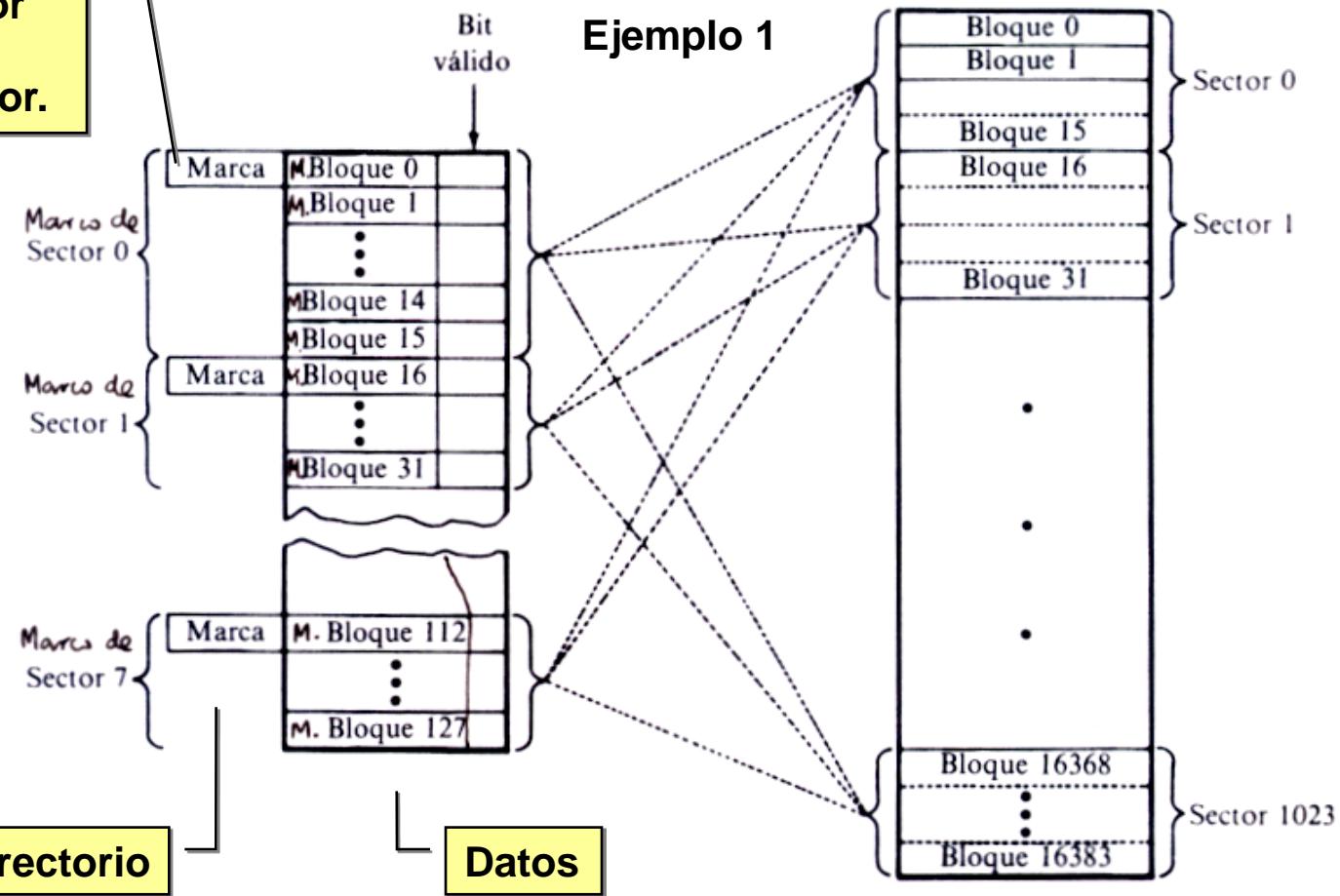
Caché: política de colocación

■ Correspondencia por sectores

- La MP se divide en 2^s grupos disjuntos llamados sectores $\Rightarrow 2^{N-s}$ bloques / sector.
- La caché se compone de $2^{n-(N-s)}$ 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 *no-válidos* al cargar un nuevo sector.
- Hay 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.

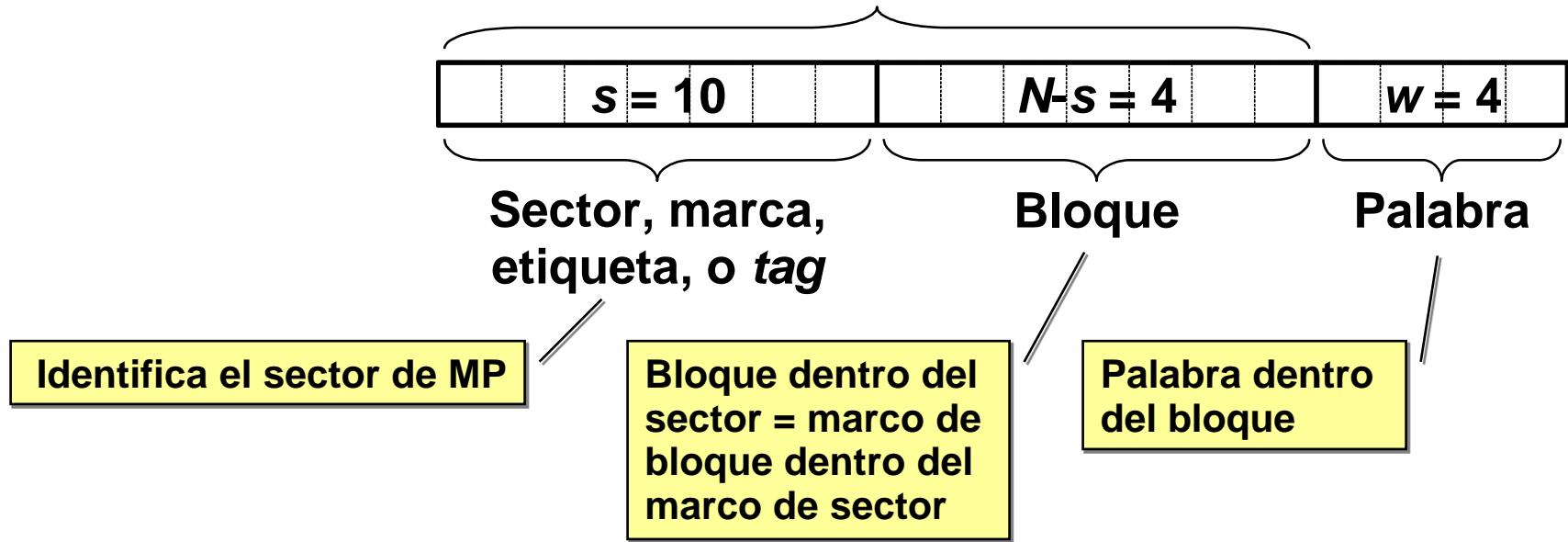
Caché: política de colocación

Cada marca contiene los s bits del sector de MP que hay en ese marco de sector.



Caché: política de colocación

- Dirección de memoria principal: $N = 14$



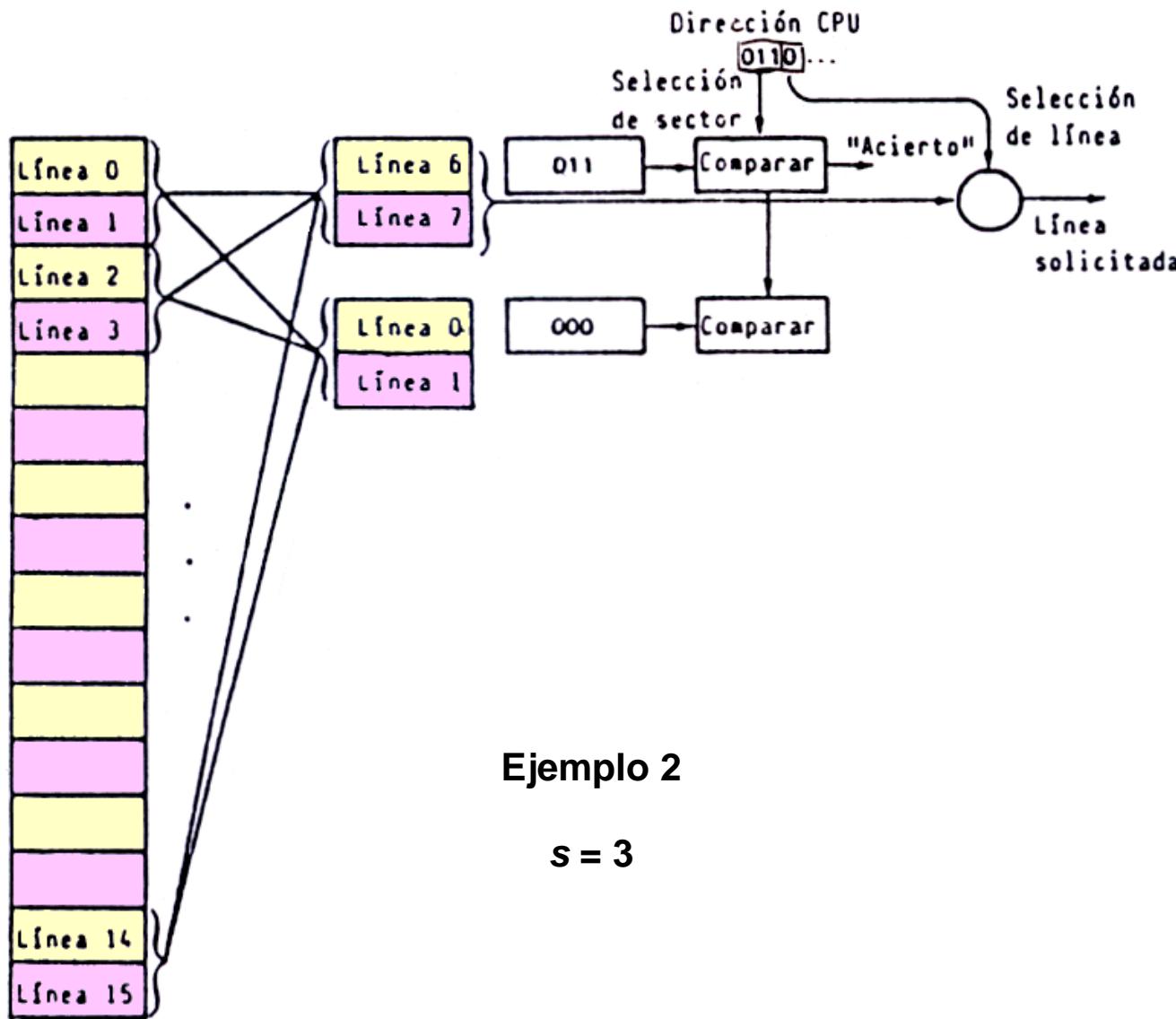
$s = 0$ (1 sector) \Rightarrow no tiene sentido (caché de igual tamaño que MP).

$s = N$ (1 bloque / sector) \Rightarrow corresp. totalmente asociativa.

✓ **Pocas marcas** \Rightarrow memoria asociativa pequeña.

✗ **Nº de combinaciones diferentes de los bloques de MP en caché es inferior al del esquema asociativo por conjuntos.**

Caché: política de colocación



Caché: política de reemplazo

■ Políticas de reemplazo de bloques

- Si se produce una falta en lectura \Rightarrow hay que traer un nuevo bloque \Rightarrow debe decidirse, si la caché está llena, qué bloque sustituir por el nuevo.
- Este problema no existe para correspondencia directa.
- Para las otras tres organizaciones:

- **ALGORITMOS DE REEMPLAZO:** ¿*en qué posición de caché se copia el bloque de MP al que se accede?*

- Se utilizan los mismos algoritmos que para memoria virtual.
 - **FIFO** (*First-In First-Out*)

» De todos los bloques existentes en caché, se reemplaza el primero que se introdujo.

- **LRU** (*Least Recently Used*)
 - » Se reemplaza el menos recientemente referenciado.
 - **RAND**
 - El bloque a reemplazar se elige aleatoriamente.

El más usado

Caché: política de actualización

■ Políticas de actualización de la memoria principal

- Cuando se modifica el contenido de la caché debe enviarse una copia de los nuevos datos a MP.
- ¿Cuándo debe realizarse esa actualización?
- Escritura directa, escribir siempre o *write-through*.
 - Se actualiza la MP cada vez que se modifica el contenido de la caché.
 - Escritura directa con asignación en escritura (EDAE):
 - Se cargan bloques en caché tanto si hay faltas por lectura como por escritura.
 - Escritura directa sin asignación en escritura (EDSAE):
 - Se cargan bloques en caché si hay faltas por lectura pero no en las faltas por escritura.
 - En general, $T(\text{EDAE}) > T(\text{EDSAE})$, ya que en la segunda se transfieren menos bloques a caché.

Caché: política de actualización

- **Post-escritura**, escribir bajo falta, o *write-back*.
 - Se actualiza la MP después, normalmente cuando ese bloque debe ser reemplazado.
 - **Post-escritura siempre** (PES):
 - Los bloques que abandonan la caché se escriben en MP.
 - Excesivo trasiego de información.
 - **Post-escritura marcada** (PEM):
 - Se añade un bit que indica si el bloque se ha modificado.
 - Se escribe en MP sólo los bloques modificados.
 - En general, $T(PES) > T(PEM)$, ya que en la segunda se transfieren menos bloques a MP.

Caché separada / unificada

■ Caches separadas de datos / instrucciones

- Es común particionar la 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.
⇒ Muchos sistemas no admiten caché de datos.
 - **Caché de instrucciones.**
 - Es fácil que bucles y pequeñas rutinas entren totalmente en caché, permitiendo su ejecución sin necesidad de acceder a MP.
 - Se simplifica la caché, ya que es habitual que se prohíba escribir en las localizaciones donde se encuentran las instrucciones ⇒ caché de **sólo lectura**.
- ✓ Se pueden emitir direcciones de instrucción y dato a la vez, doblando el ancho de banda entre caché y procesador.
- ✓ Se puede optimizar cada caché por separado:
 - Diferentes capacidades, tamaños de bloque, asociatividades, etc.

Caché separada / unificada

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

¿Cuál tiene una frecuencia de fallos menos: una caché de instrucciones de 16 KB + una caché de datos de 16 KB o una caché unificada de 32 KB?

Frecuencia de fallos para la caché particionada:

$$53\% \cdot 3,6\% + 47\% \cdot 5,3\% = 4,4\%$$

Una caché unificada de 32 KB tiene una frecuencia de fallos similar (4,3%).

(Ver, sin embargo, las ventajas de la transparencia anterior).

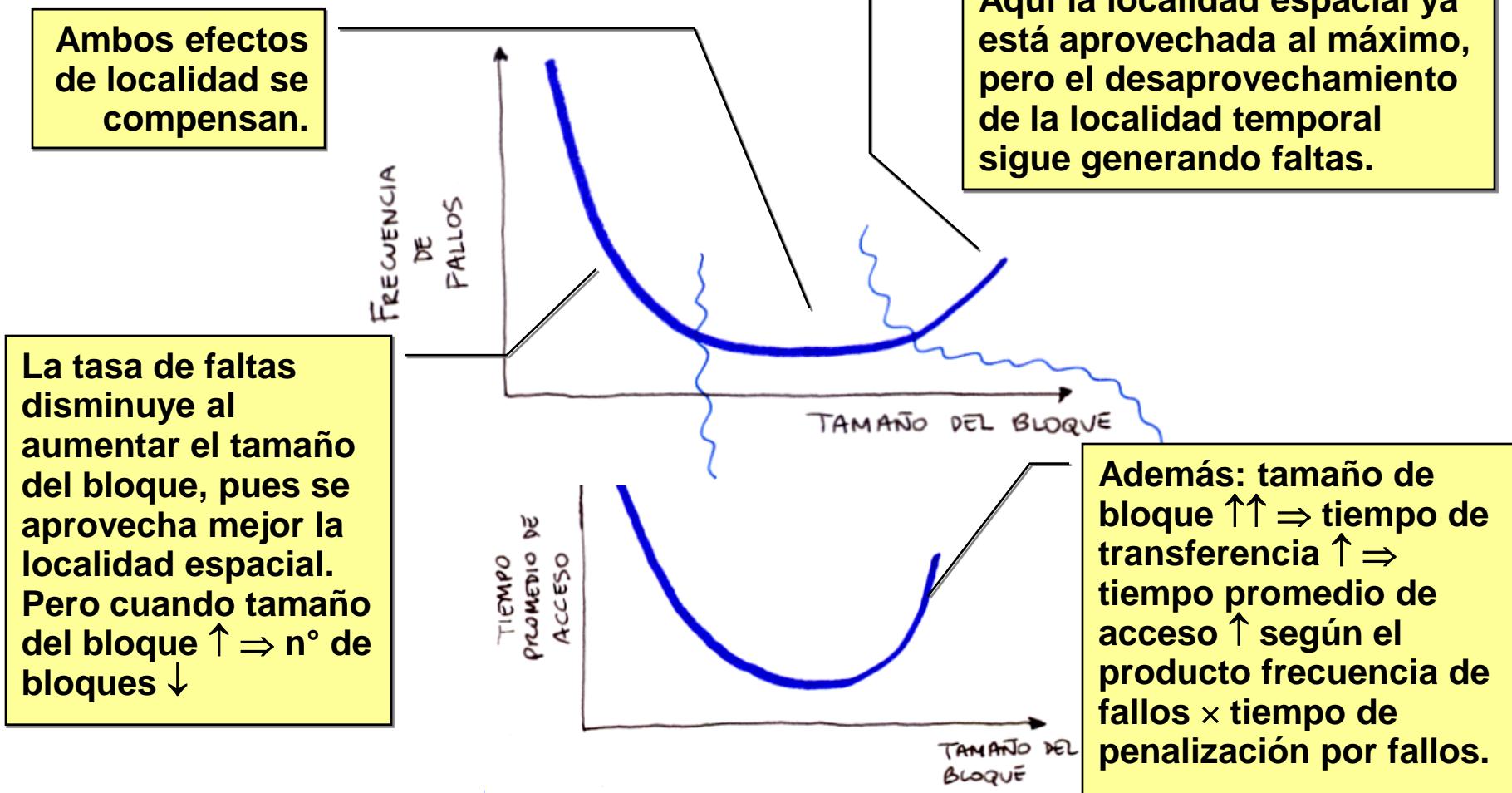
Tamaño	Instrucción sólo	Sólo datos	Unificada
0,25 KB	22,2 %	26,8 %	28,6 %
0,50 KB	17,9 %	20,9 %	23,9 %
1 KB	14,3 %	16,0 %	19,0 %
2 KB	11,6 %	11,8 %	14,9 %
4 KB	8,6 %	8,7 %	11,2 %
8 KB	5,8 %	6,8 %	8,3 %
16 KB	3,6 %	5,3 %	5,9 %
32 KB	2,2 %	4,0 %	4,3 %
64 KB	1,4 %	2,8 %	2,9 %
128 KB	1,0 %	2,1 %	1,9 %
256 KB	0,9 %	1,9 %	1,6 %

Frecuencia de fallos para caches de distintos tamaños de sólo datos, sólo instrucciones, y unificadas. Los datos son para una cache 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 del 53 por 100.

Caché: tamaño

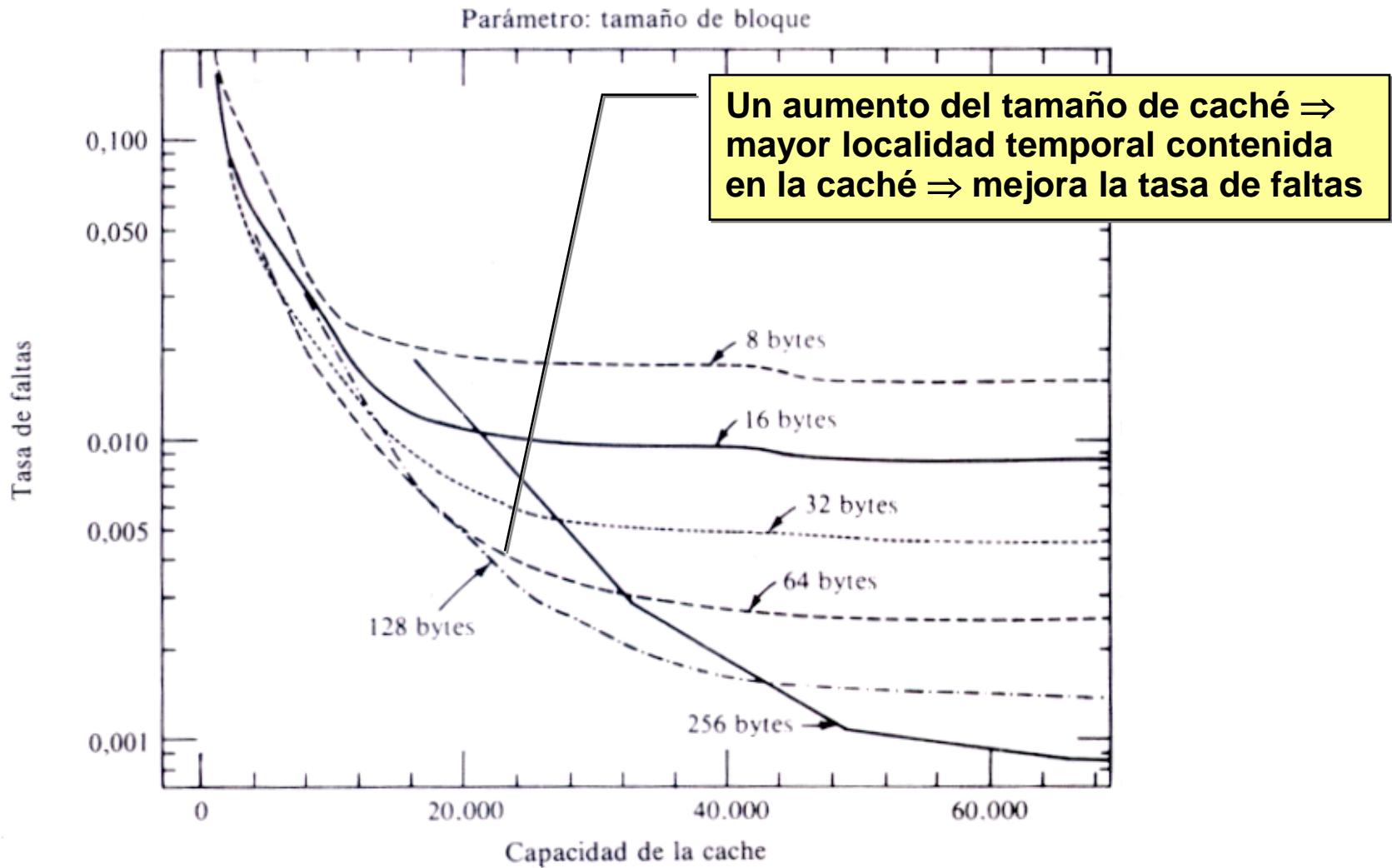
■ Tamaños del bloque y de la caché

- Para un tamaño de caché fijo:



Caché: tamaño

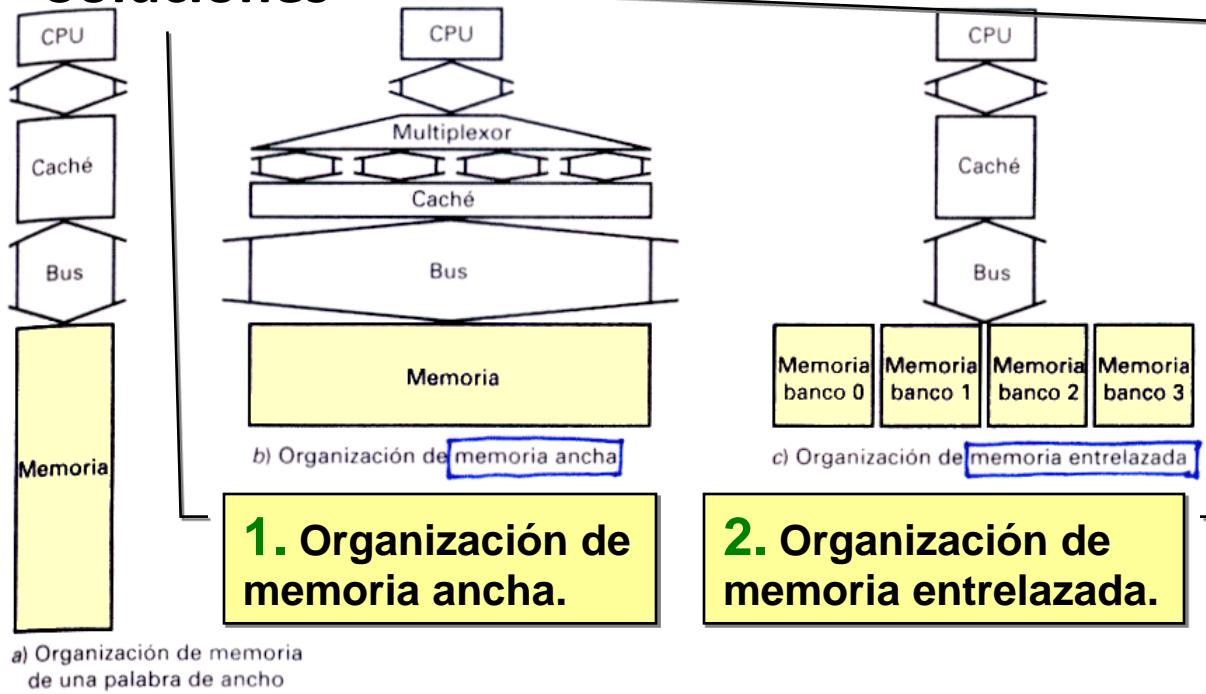
- Para un tamaño de bloque fijo:



Diseño para dar soporte a caché

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 MP y caché.

Soluciones



3. Acceso a la memoria en modo página rápido, o utilizando algunos de los tipos de memoria DRAM más recientes (SDRAM, RDRAM).

1. Organización de memoria ancha.

2. Organización de memoria entrelazada.

a) Organización de memoria de una palabra de ancho

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 entrelazados.

Diseño para dar soporte a caché

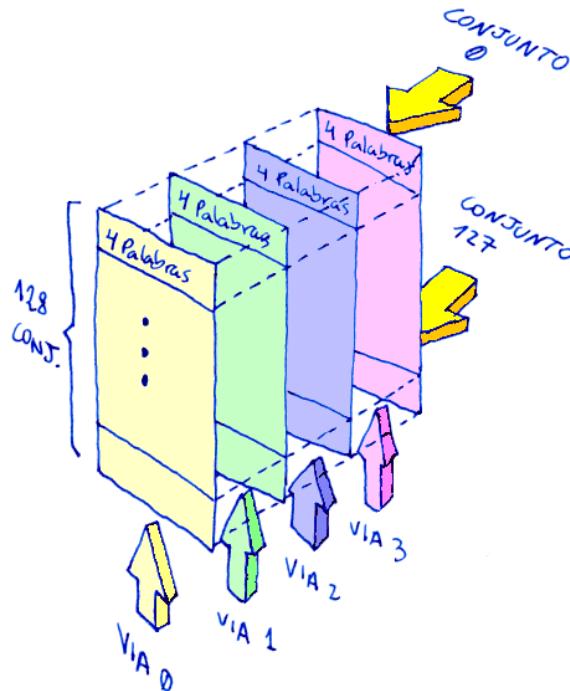
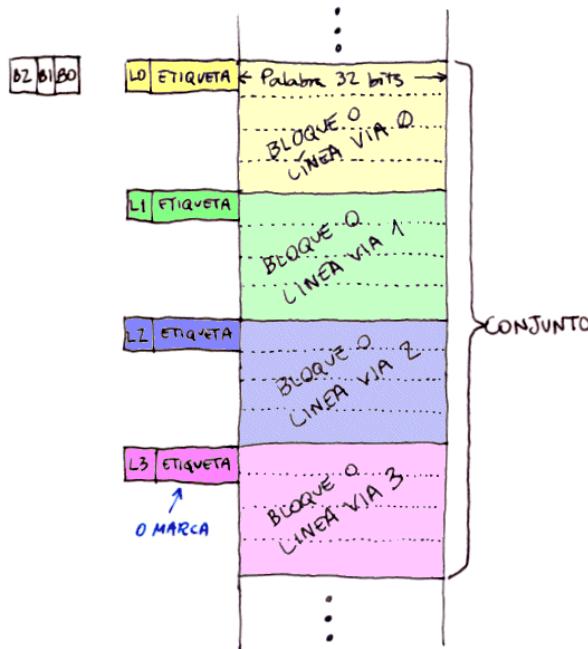
■ Ejemplo (transferencia de 4 palabras de MP a caché):

- 1 ciclo de reloj para enviar la dirección.
- 10 ciclos para cada el acceso a la DRAM.
- 1 ciclo para enviar una palabra de datos.
- Organización de una palabra de ancho:
 - $1 + 4 * 10 + 4 * 1 = 45$ ciclos
- Organización de memoria ancha (2 palabras):
 - $1 + 2 * 10 + 2 * 1 = 23$ ciclos
- Organización de memoria ancha (4 palabras):
 - $1 + 1 * 10 + 1 * 1 = 12$ ciclos
- Organización de memoria entrelazada (4 módulos):
 - $1 + 1 * 10 + 4 * 1 = 15$ ciclos

Ejemplo: caché interna del 486

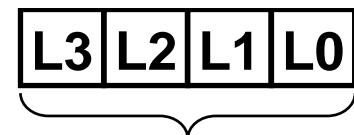
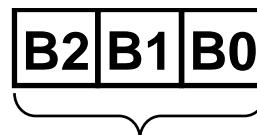
■ Ejemplo: Unidad de caché interna del i486

- Caché unificada (datos e instrucciones) de 8 KB.
- Correspondencia asociativa por conjuntos, 128 conjuntos
- Cada conjunto tiene 4 bloques o líneas \Rightarrow asociativa de 4 vías.
- Cada bloque tiene 4 palabras de 32 bits (16 bytes).



Ejemplo: caché interna del 486

- Cada marco de bloque tiene una **etiqueta o marca de 21 bits**, que se compara con los 21 bits de mayor peso ($A_{11}-A_{31}$) de la dirección física referenciada para ver si está en caché.
- Cada conjunto: **7 bits de estado**:



- **Algoritmo de extracción:**

- **Por demanda selectiva.**

- El llenado de una línea de caché se lleva a cabo siempre que se produzca una falta en lectura. Si la falta es en escritura, la caché no intentará cargar el bloque.

Utilizados por el algoritmo de reemplazo LRU

Indican qué líneas son válidas (contienen datos correctos)

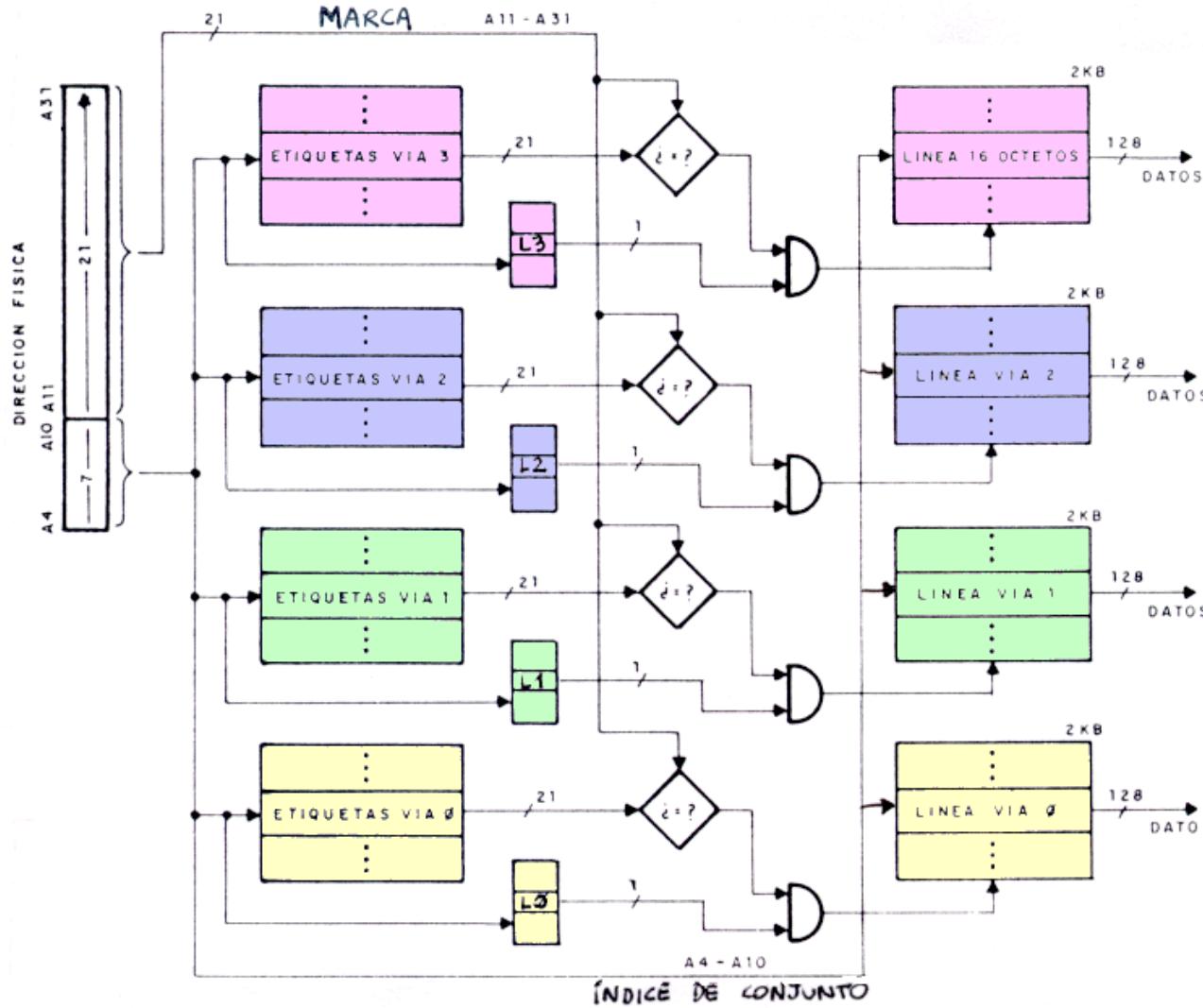
- **Actualización de la memoria principal:**

- **Escritura directa sin asignación en escritura.**

- La escritura sobre un dato contenido en la caché es inmediatamente dirigida hacia MP pasando antes por un registro intermedio de escritura (el i486 no tiene que esperar a que finalice el ciclo de escritura).

Ejemplo: caché interno del 486

■ Organización física



Ejemplo: caché interna del 486

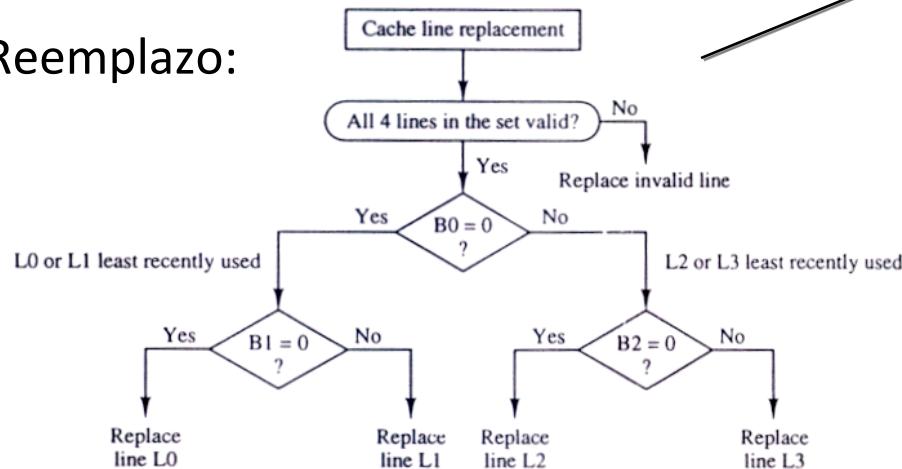
- Algoritmo de reemplazo:

- **Pseudo-LRU** (no es LRU; ej.: L2 L1 L0 L3):

- Modificación de los bits B0, B1 y B2:

```
if (acceso a L0 o a L1)
{
    B0 = 1;
    if (acceso a L0) B1 = 1;
    else B1 = 0;
}
else
{
    B0 = 0;
    if (acceso a L2) B2 = 1;
    else B2 = 0;
}
```

- Reemplazo:



Acceso a	B0	B1	B2
L0	1	1	
L1	1	0	
L2	0		1
L3	0		0

Sólo se utiliza el algoritmo de reemplazo si todas las líneas del conjunto son válidas. Si no, los nuevos datos se almacenan en alguna de las no válidas.