Organización del Procesador

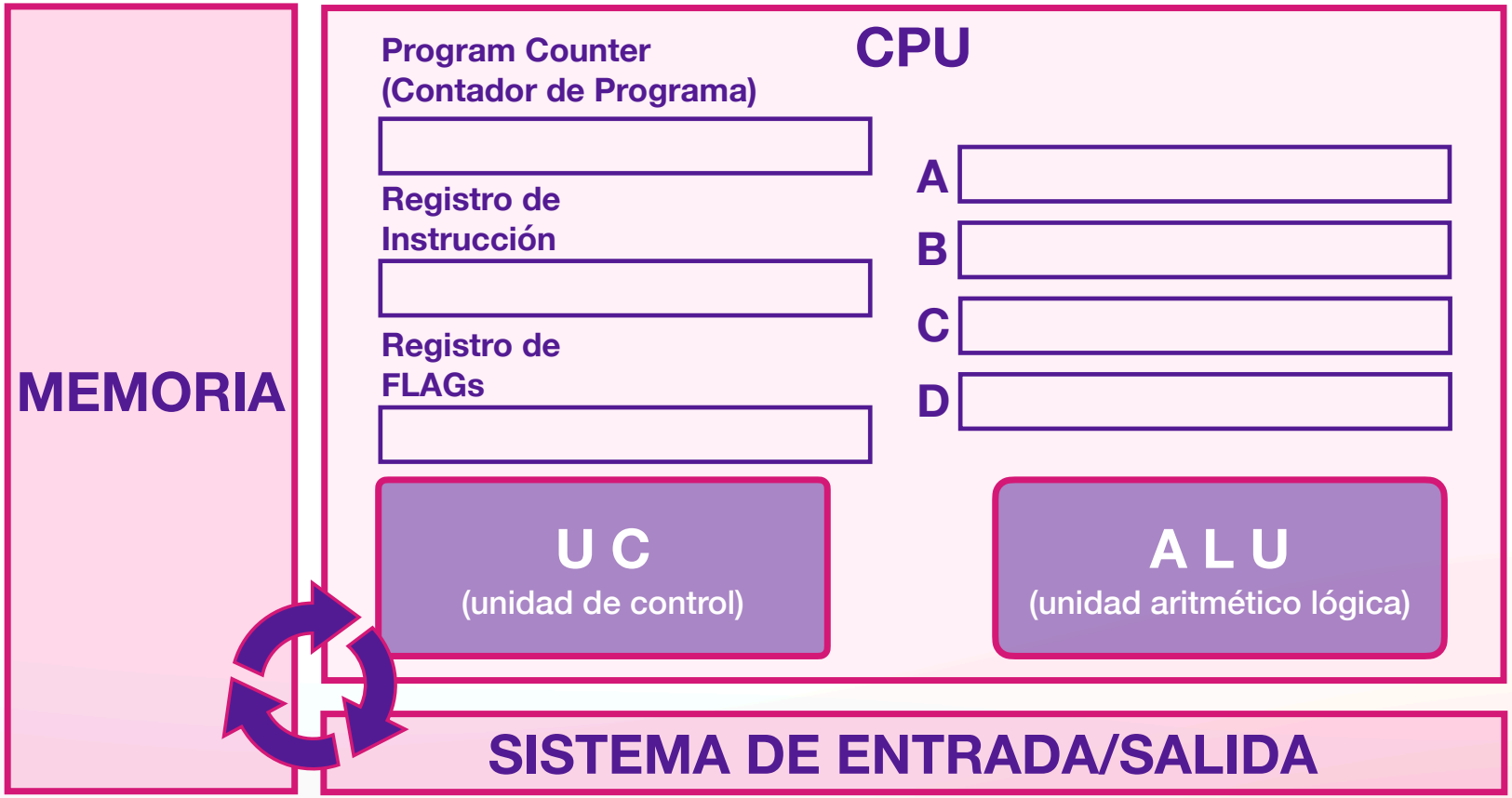
Memoria (Hardware - Cache - Virtual)

Departamento de Computación - UNRC

El camino a recorrer

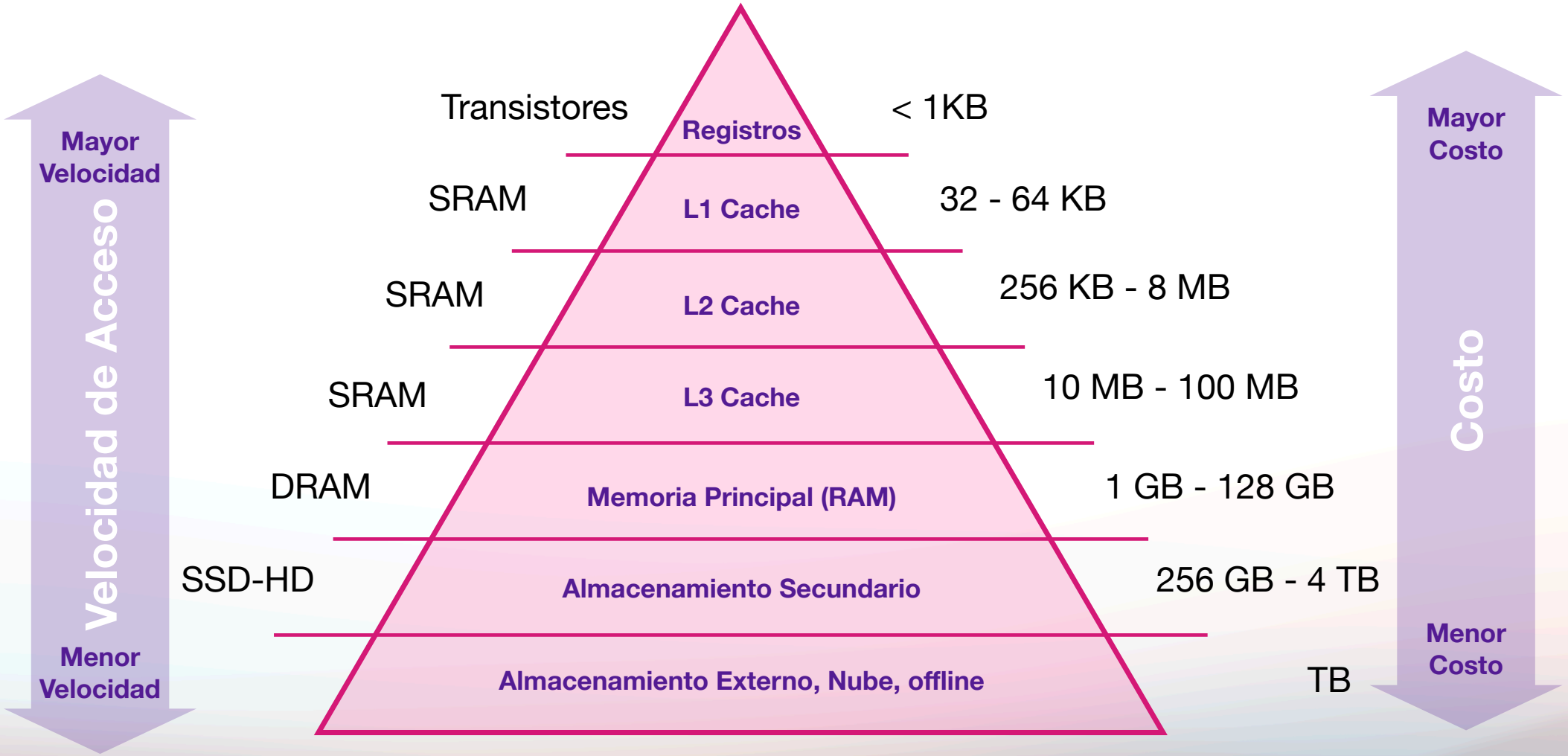
- Un poco de Historia y Sistemas Numéricos
- Introducción a la Electrónica
- Representación de Información
- Cómo computar utilizando la electricidad
- Evolución y funcionamiento abstracto de una computadora
- Assembly X86
- Micro-programación (cómo fabricar un procesador)
- Eficiencia
  - Pipelines
  - **Memoria (Hardware)**
  - **Memoria (Cache)**
  - **Memoria (Virtual)**

Arquitectura von Neumann



John von Neumann: (1903-1957) Físico Matemático Húngaro, Estadounidense. Participó activamente en el proyecto ENIAC y en el desarrollo de la EDVAC, donde propuso el concepto de almacenamiento de programas y datos en la memoria de la computadora, sentando las bases de la arquitectura de von Neumann, que se convirtió en el modelo predominante en las computadoras modernas.

Jerarquía de Memorias



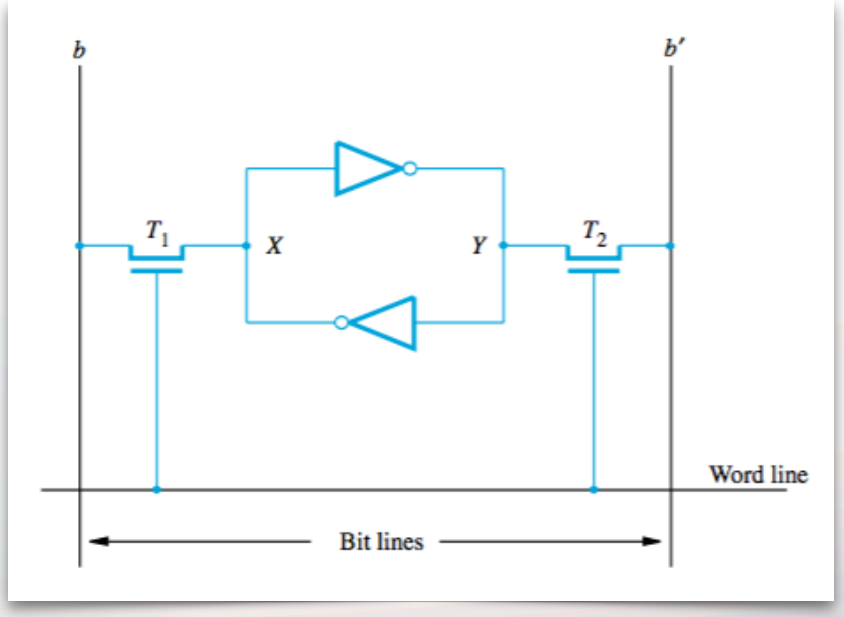
Tipos de Memorias

- **RAM (Memoria de Acceso Directo):** Comunmente denominada memoria principal. Es la memoria (lectura/escritura) con la que trabajamos cuando programamos, en la cual residen los programa y datos. Es **volátil** (sus datos solo se mantiene mientras tienen energía eléctrica). Si bien existe muchas variantes, podemos clasificarlas según su composición física en **estáticas** SRAM y **dinámicas** DRAM.

- **ROM (Memoria de sólo Lectura):** A diferencia de las RAM, sus datos **persisten aún sin energía eléctrica**. En genera se utilizan en pequeñas unidades conteniendo información (programas) indispensables para el funcionamiento, como el arranque, configuración de dispositivos, software embebido, etc. Dependiendo de su conformación y modo de grabado existen PROM, EPROM, EEPROM …

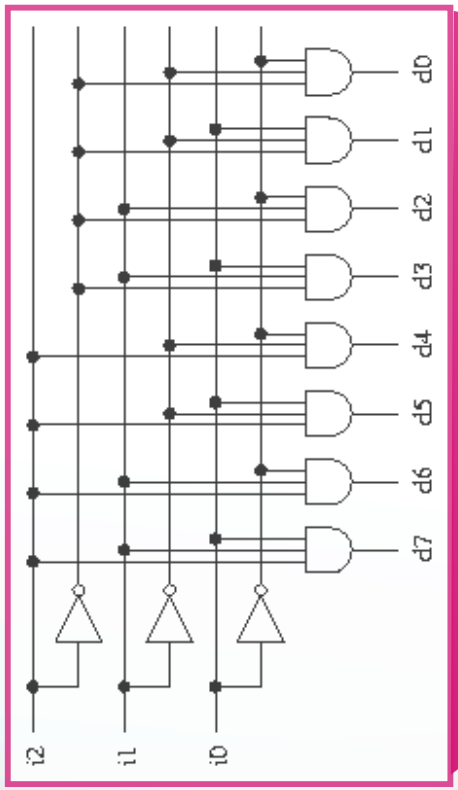
Memorias SRAM

- **SRAM (Static RAM)**: Está compuesta por circuitos similares a los Flip-Flop. Son memorias de acceso rápido pero su construcción es costosa. Consumen más energía y por lo tanto disipan más calor.

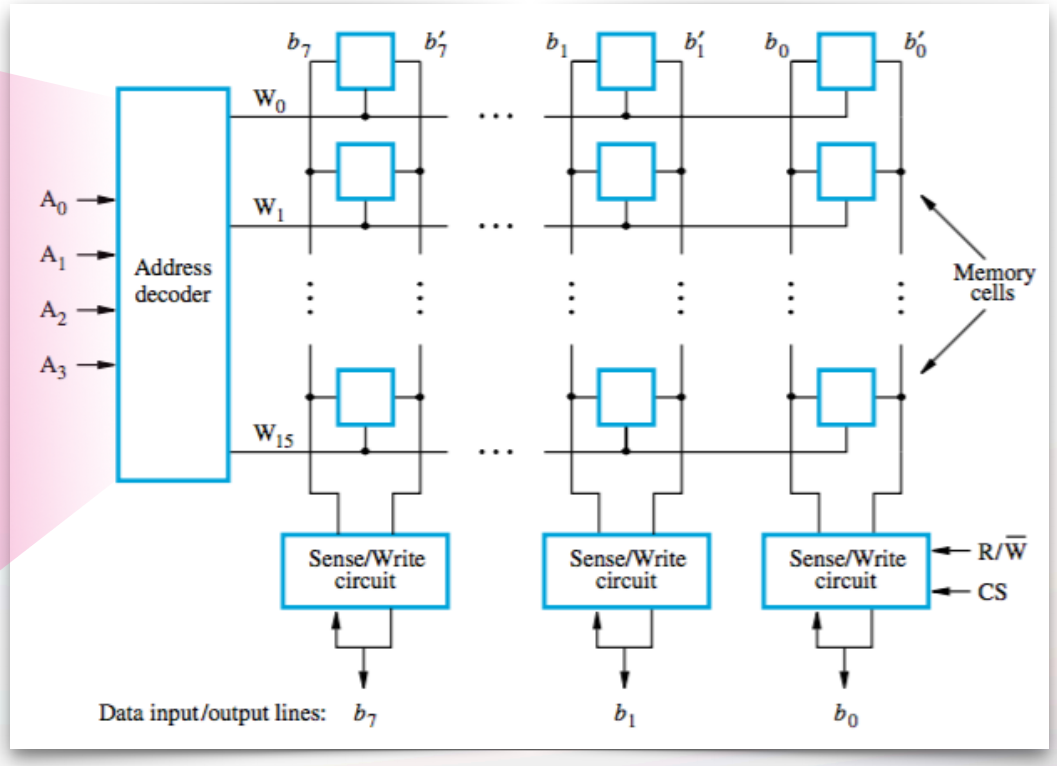


Memorias SRAM

- **SRAM (Static RAM)**: Ejemplo de una organización 128B x 8

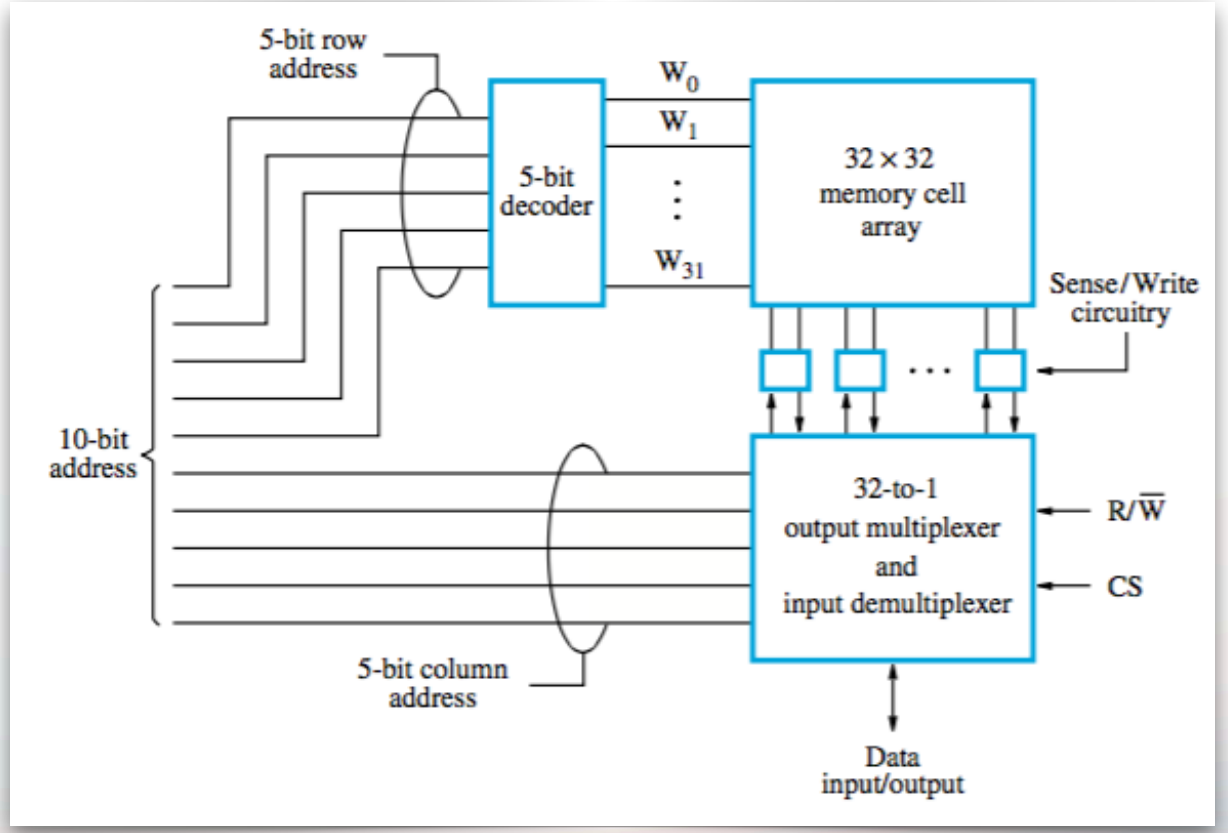


Ejemplo de Decodificador de 3 bits a 8 líneas



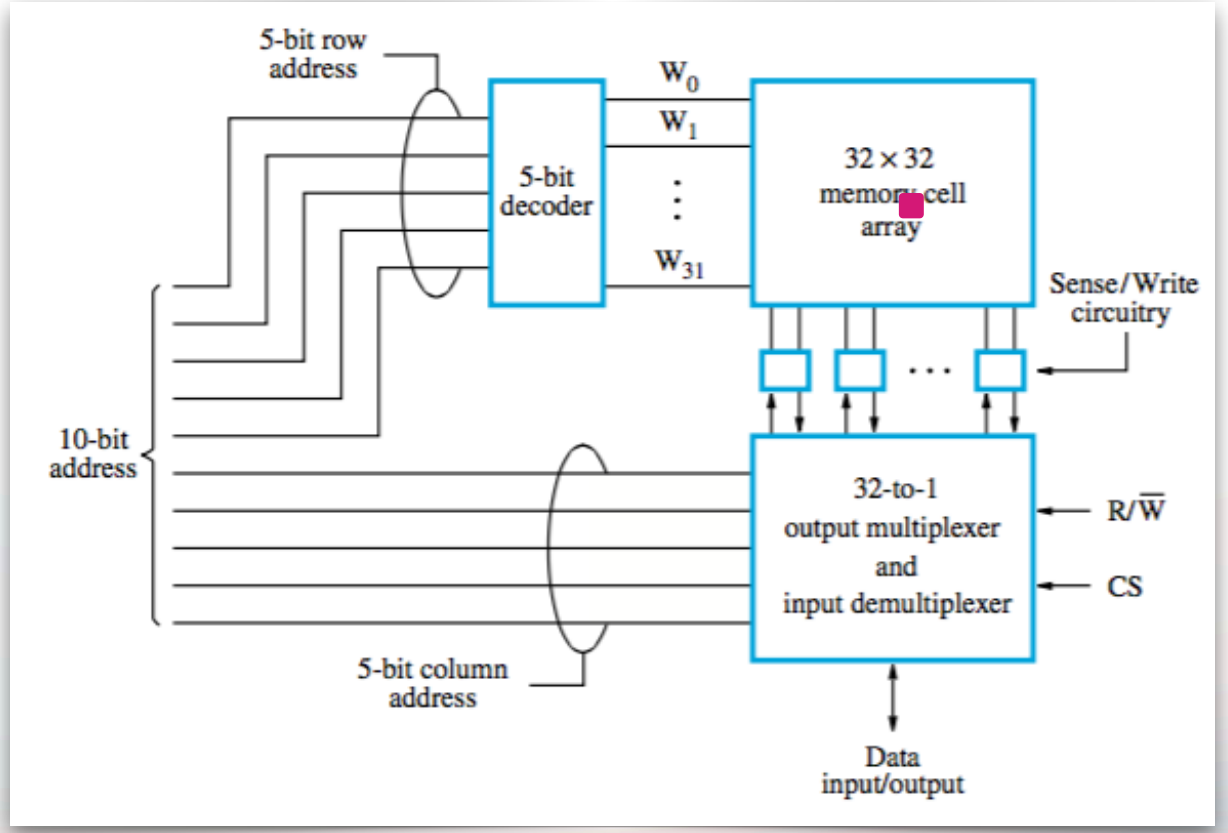
Memorias SRAM

- **SRAM (Static RAM)**: Ejemplo de una organización 1024 (1K) x 1



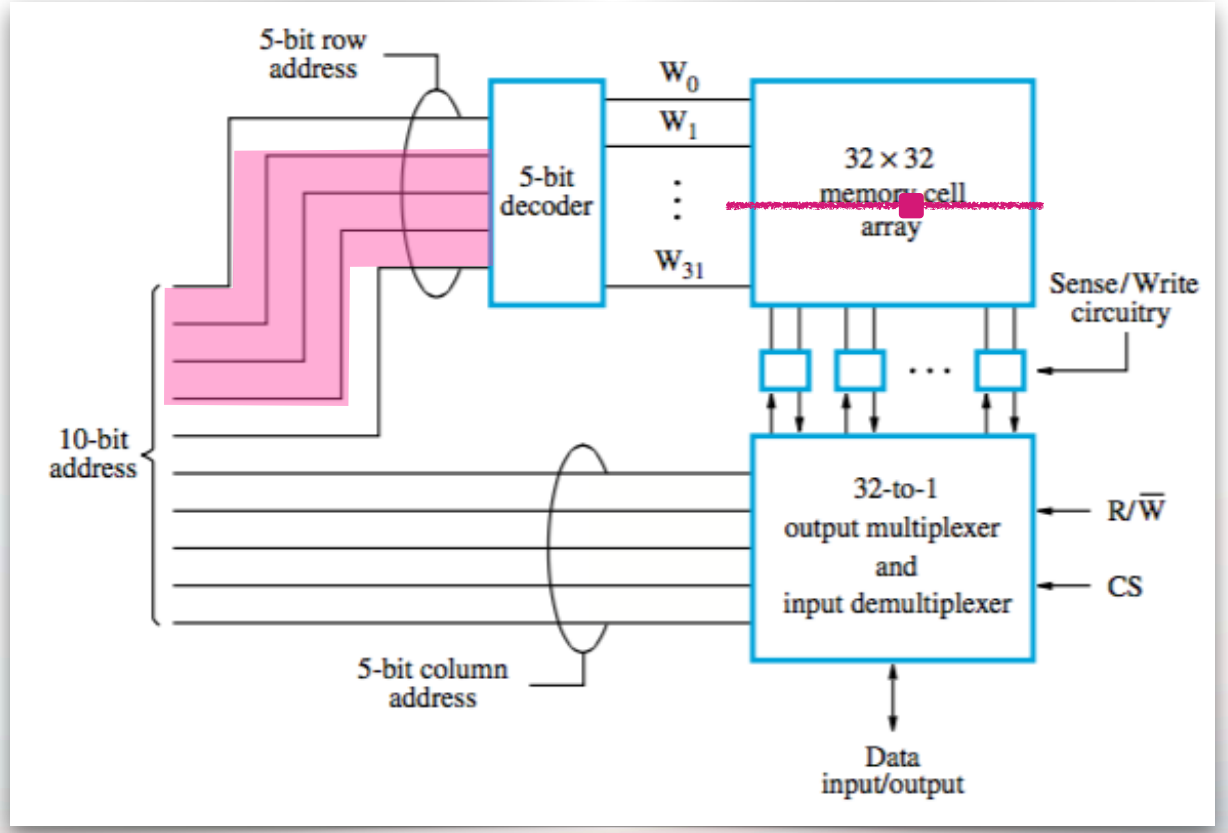
Memorias SRAM

- **SRAM (Static RAM)**: Ejemplo de una organización 1024 (1K) x 1



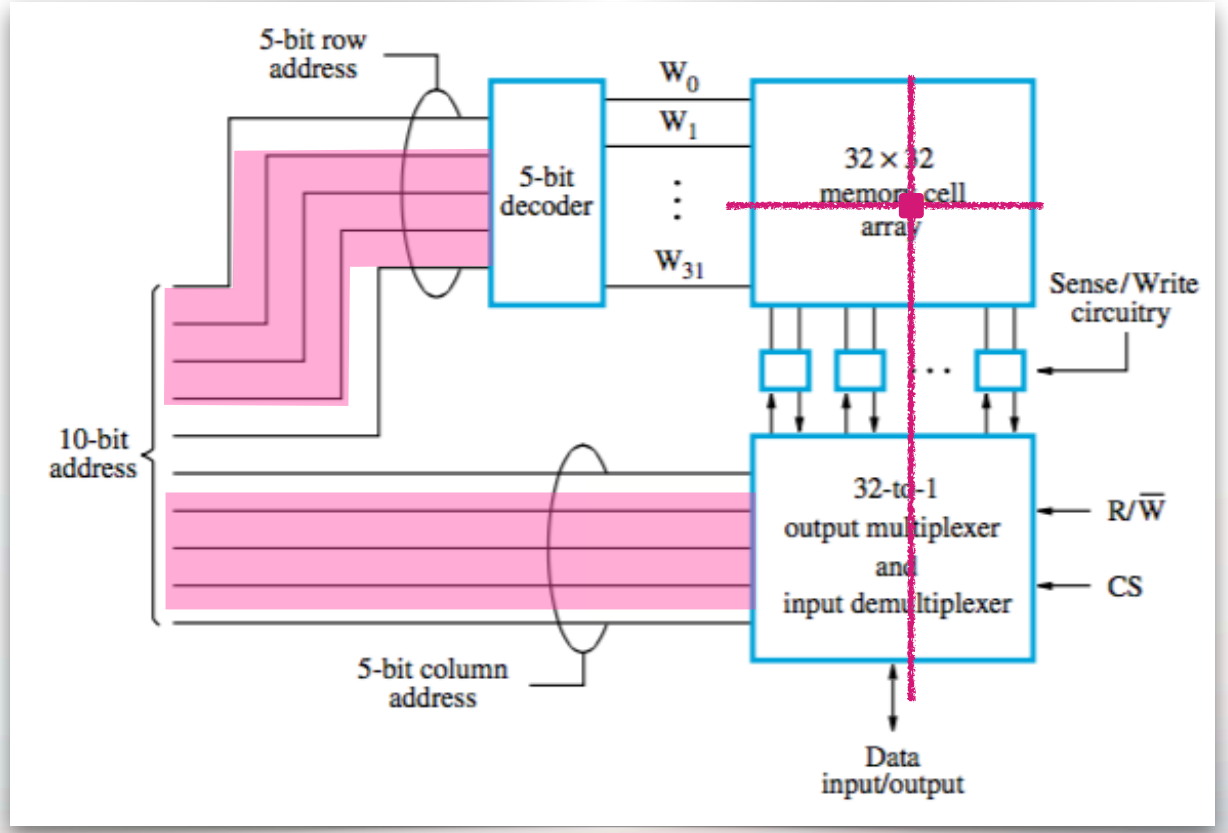
Memorias SRAM

- **SRAM (Static RAM)**: Ejemplo de una organización 1024 (1K) x 1



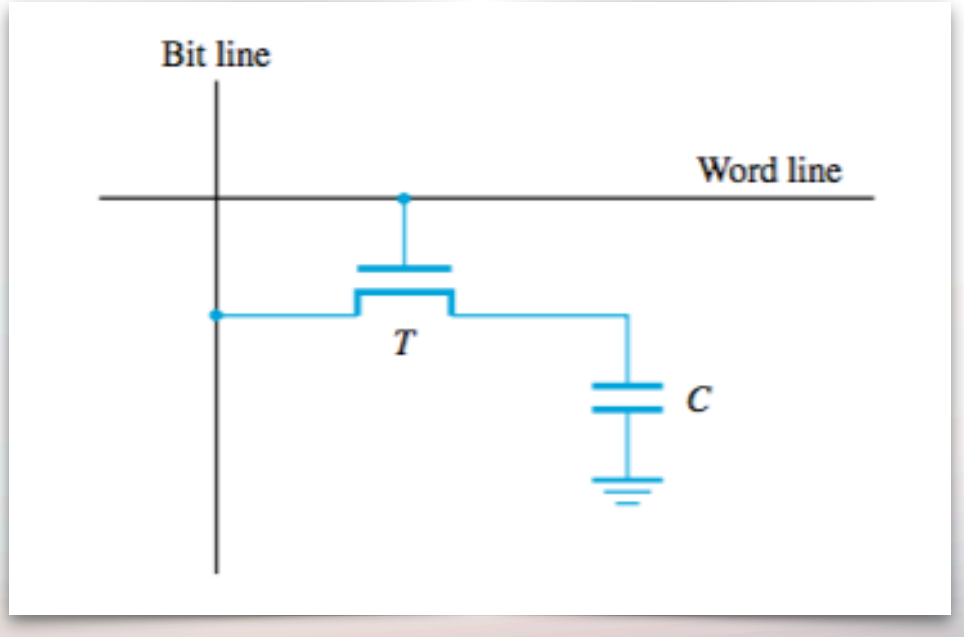
Memorias SRAM

- **SRAM (Static RAM)**: Ejemplo de una organización 1024 (1K) x 1



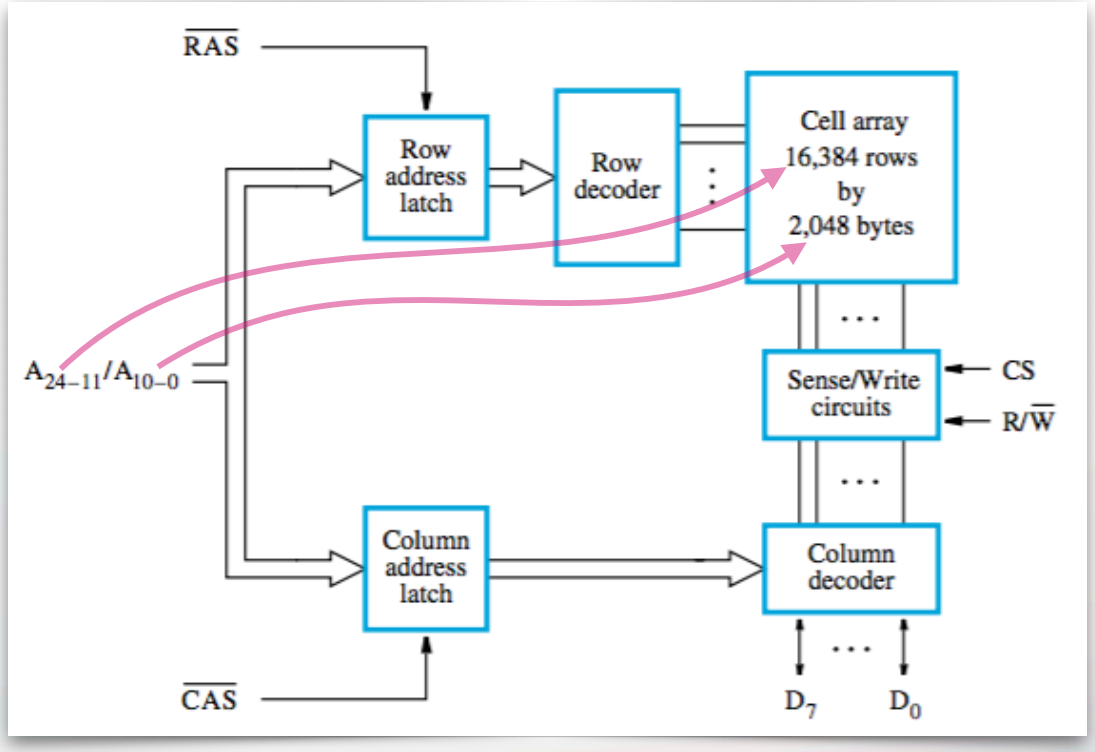
Memorias DRAM

- **DRAM (Dynamic RAM)**: La celdas de memoria dinámica utilizan capacitores en lugar de transistores, son más baratas, consumen menos energía. Otra diferencia es que su valor deber ser revalidado (refresh).



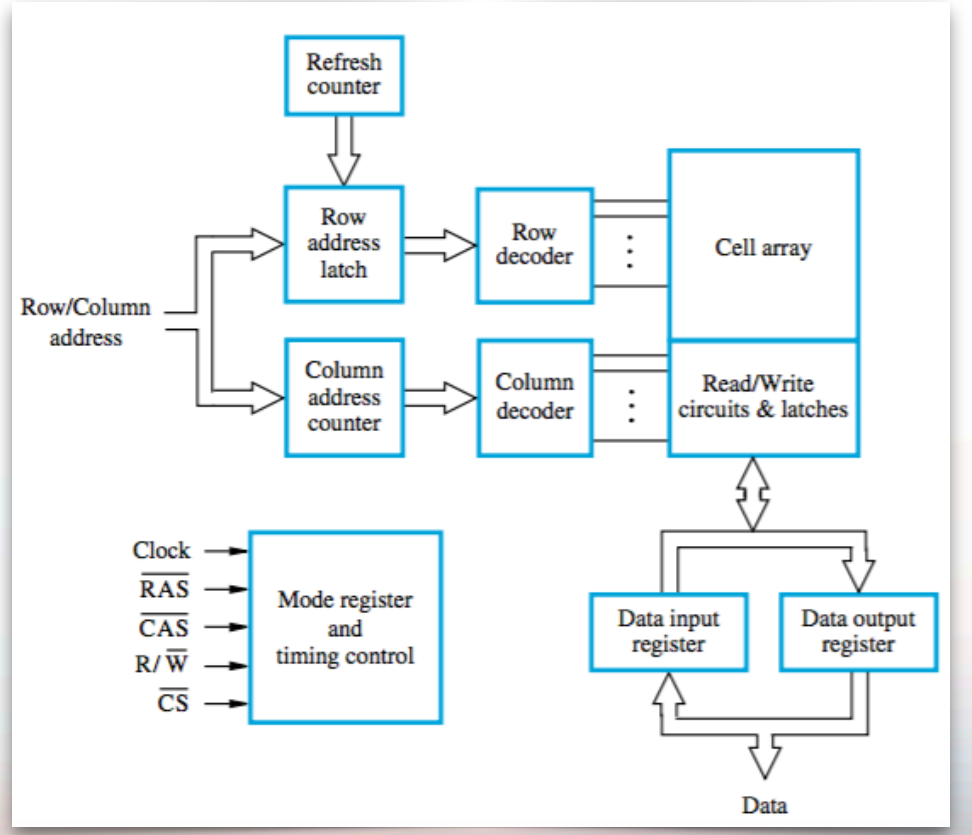
Memorias DRAM

- **DRAM**: Ejemplo de una organización para 32 MBytes



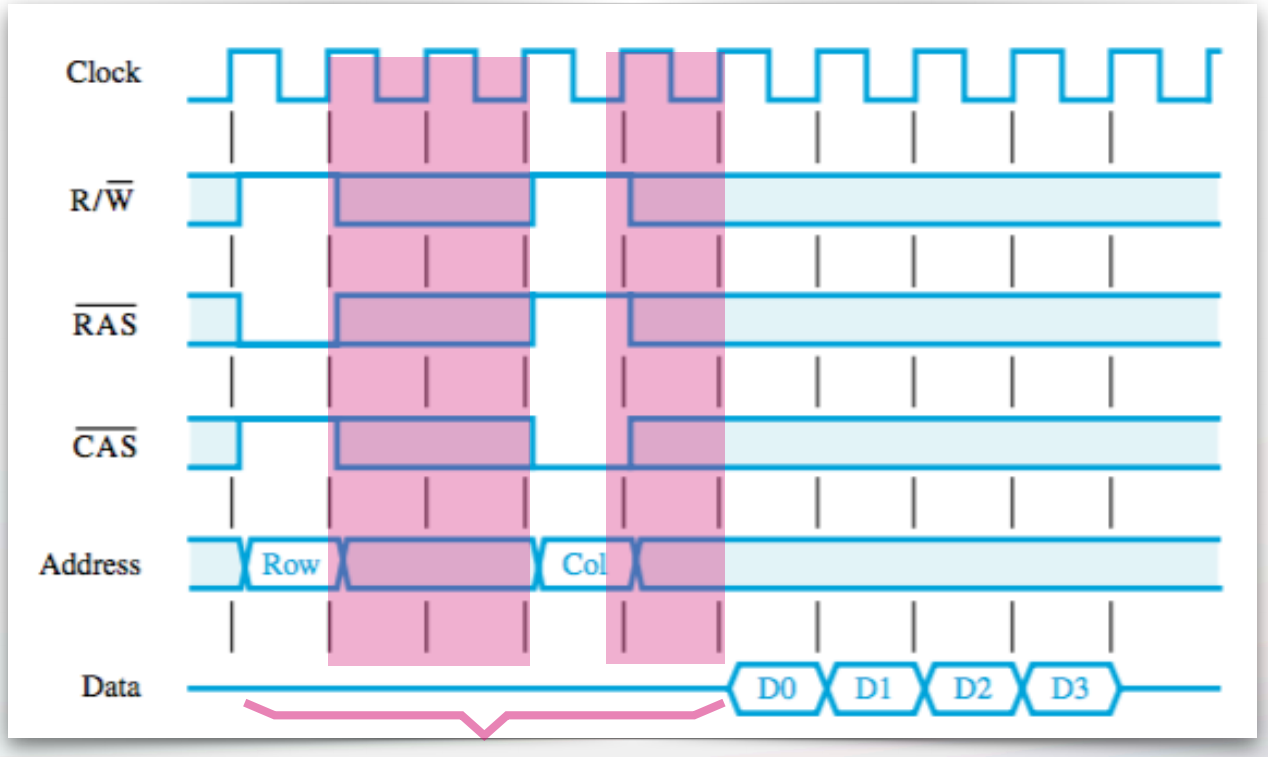
Memorias SDRAM

- **SDRAM (Synchronous Dynamic RAM):** A diferencia de las DRAM, la memoria SDRAM está sincronizada mediante un clock. Esto le otorga mayor control dentro del chip.



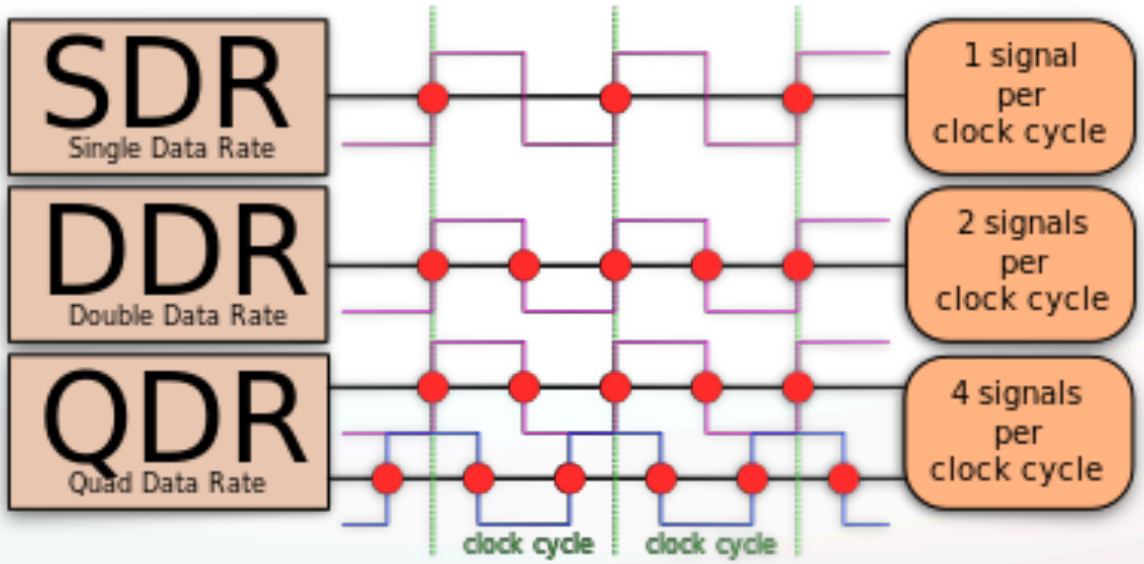
Memorias SDRAM

- **SDRAM (Synchronous Dynamic RAM):** Tiempos de acceso y latencia



Memorias SDRAM

- **DDR (Double Data Rate):** Técnica de acceso para mejorar la eficiencia



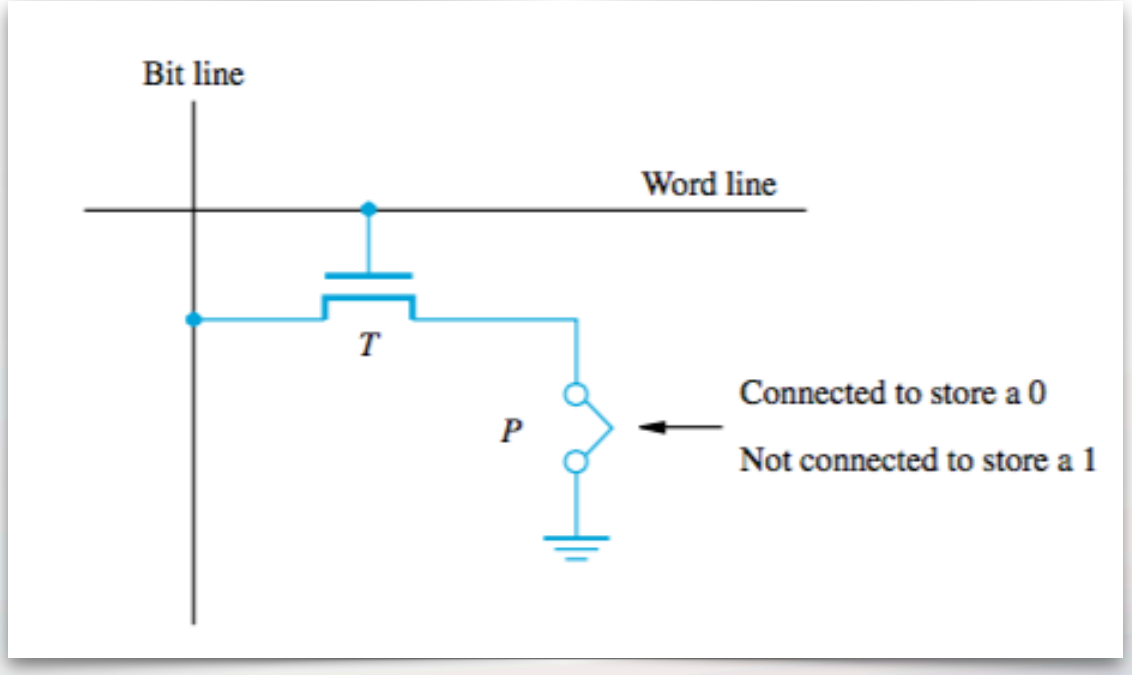
Memoria principal - Latencia y Bandwidth

- **Latencia**: Tiempo en obtener el primer Página de datos. Usualmente se mide de manera relativo como cantidad de ciclos de reloj. Por ej. si es de 5 ciclos y el reloj es de 500MHz, entonces el tiempo es de 10 ns.
- **Bandwidth**: Cantidad de datos (bits o Bytes) que la memoria pues transferir por segundo.

Nombre	Reloj E/S	Promedio Transferencia	Bandwith Teórico
DDR-200, PC-1600	100 MHz	0.2 GB/s	1.6 GB/s
DDR2-800, PC2-6400	400 MHz	0.8 GB/s	6.4 GB/s
DDR3-1600, PC3-12800	800 MHz	1.6 GB/s	12.8 GB/s
DDR4-3200, PC4-25600	1600 MHz	3.2 GB/s	25.6 GB/s
DDR5-6400, PC5-51200	3200 MHz	6.4 GB/s	51.2 GB/s

Memoria ROM

- **ROM (Read Only Memory)**: La celdas de memoria no pueden modificarse (durante su utilización).



Tipos memorias ROM

- **PROM** (Programable ROM): El material conector son fusibles, el proceso de grabado (quema) los de las celdas a contener 1.

- **EPROM** (Erasable Programable ROM): A diferencia de las PROM, tiene transistores en vez de fusibles, estos puede ser borrados y vueltos a programa mediante luz ultravioleta.

- **EEPROM** (Electrical Erasable Programable ROM): Son memorias reprogramables con electricidad, a diferencia de las EPROM, utilizan diferentes voltajes para activar o desactivar los transistores de las celdas.

- **FLASH**: En esencia son EEPROM pero que sólo pueden ser modificados de a Páginas reduciendo la complejidad de circuitos y su costo.

Organización del Procesador

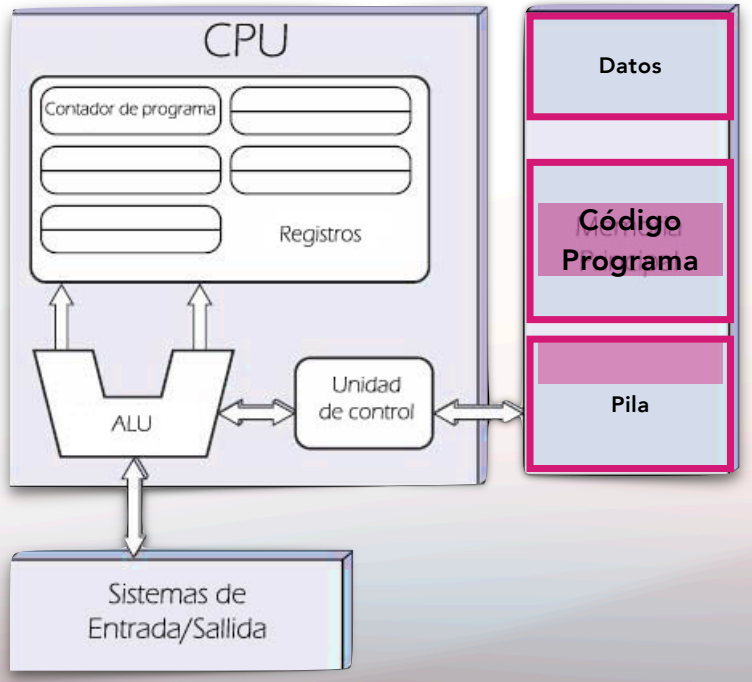
Memoria Cache

Departamento de Computación - UNRC

Memoria Cache - Principio de LOCALIDAD

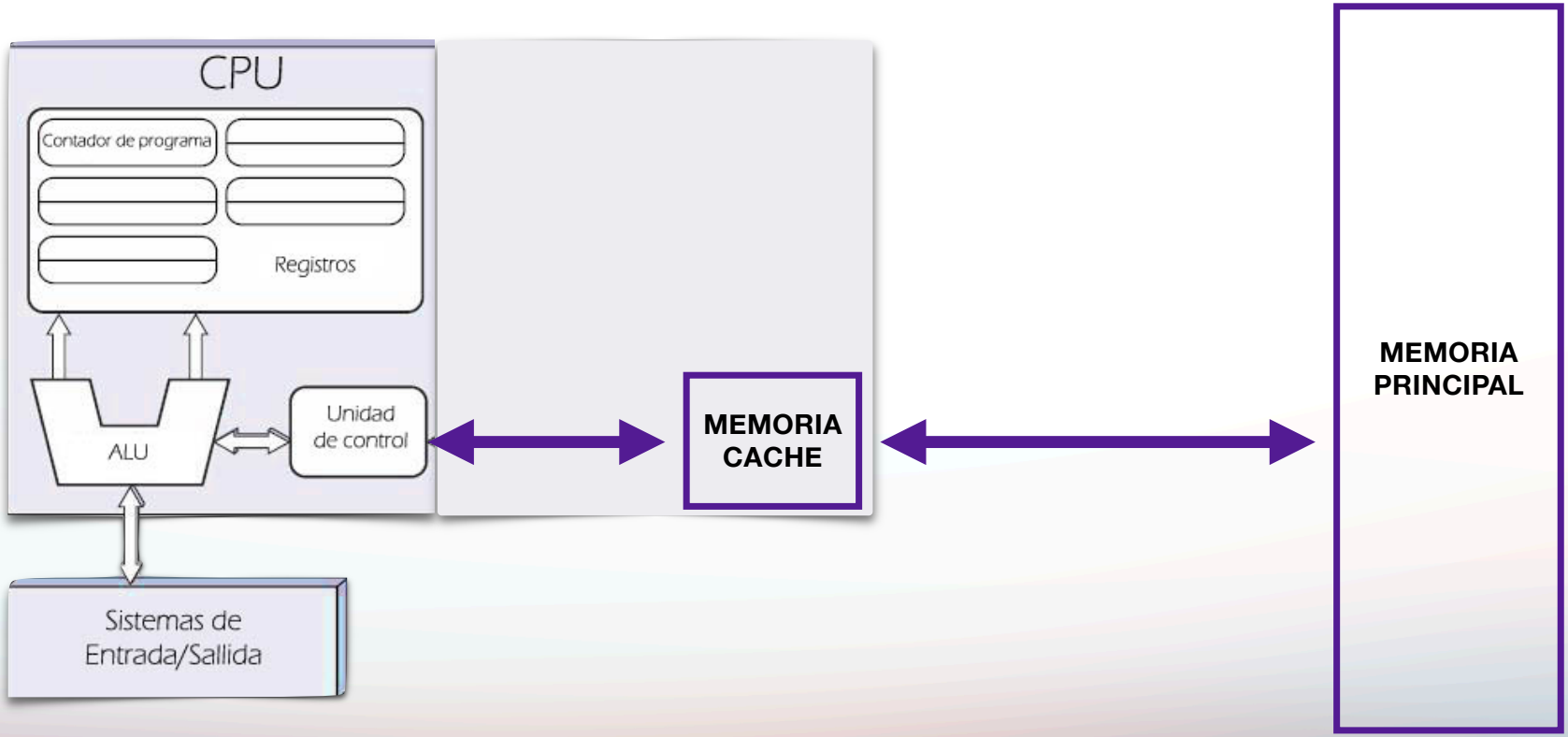
En general el acceso a memoria presenta un patrón bastante frecuente, si se accede a una posición X, es altamente probable que luego se aceda a la posición X+1 en caso de programas, o en posiciones cercanas, por ejemplo en la Pila. A este patrón de utilización se lo denomina localidad (proximidad).

- **Temporal**: datos utilizados se volverán a utilizar en el futuro cercano, por ej. variables locales.
- **Espacial**: datos cercanos (ubicación en memoria), por ej. arreglos.
- **Secuencial**: las instrucciones de programas en genera tienen un comportamiento de uso secuencia



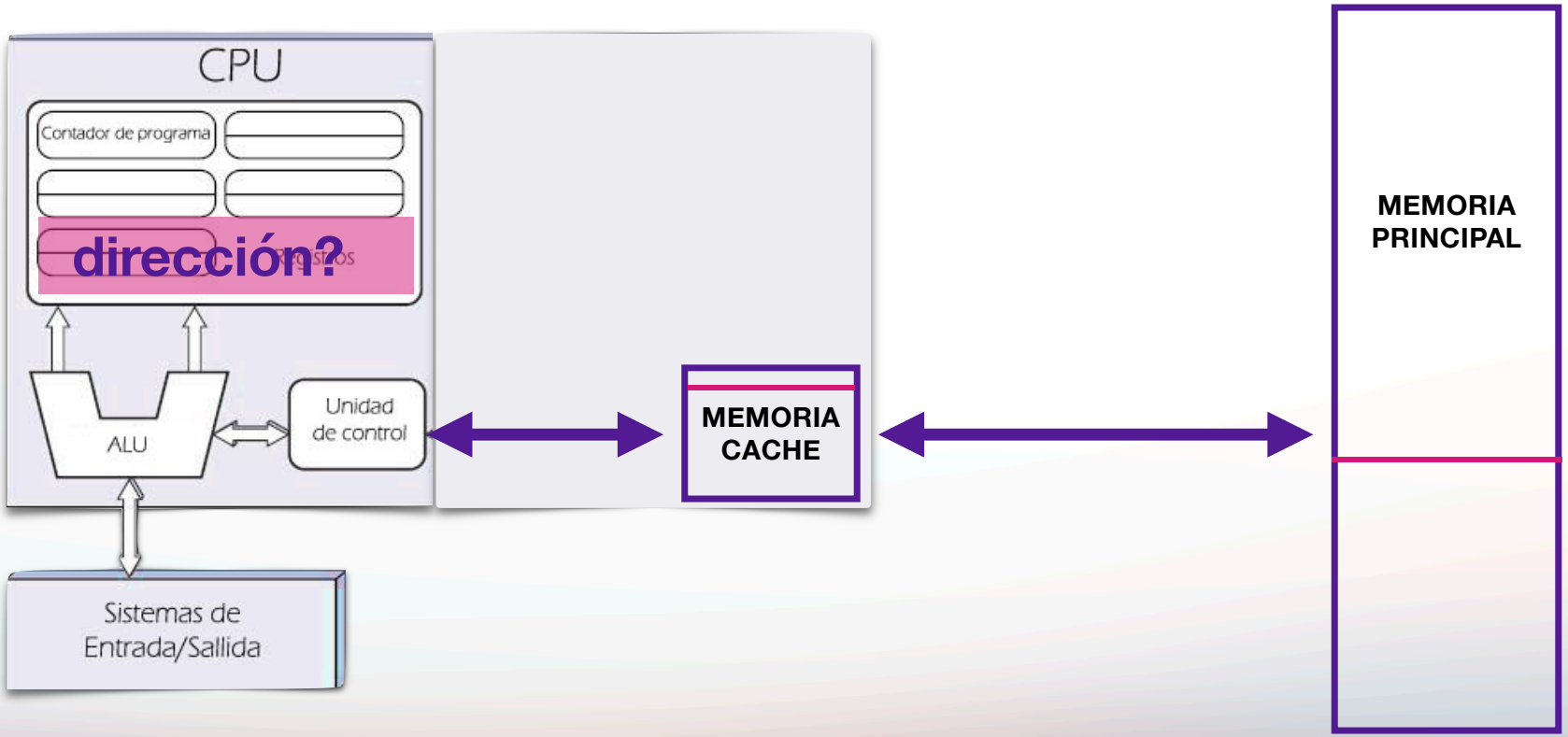
Memoria Cache

El patrón de localidad permite ganar mucha eficiencia trabajando de a **bloques** de memoria. Para ello se utiliza la idea de **memoria cache**.



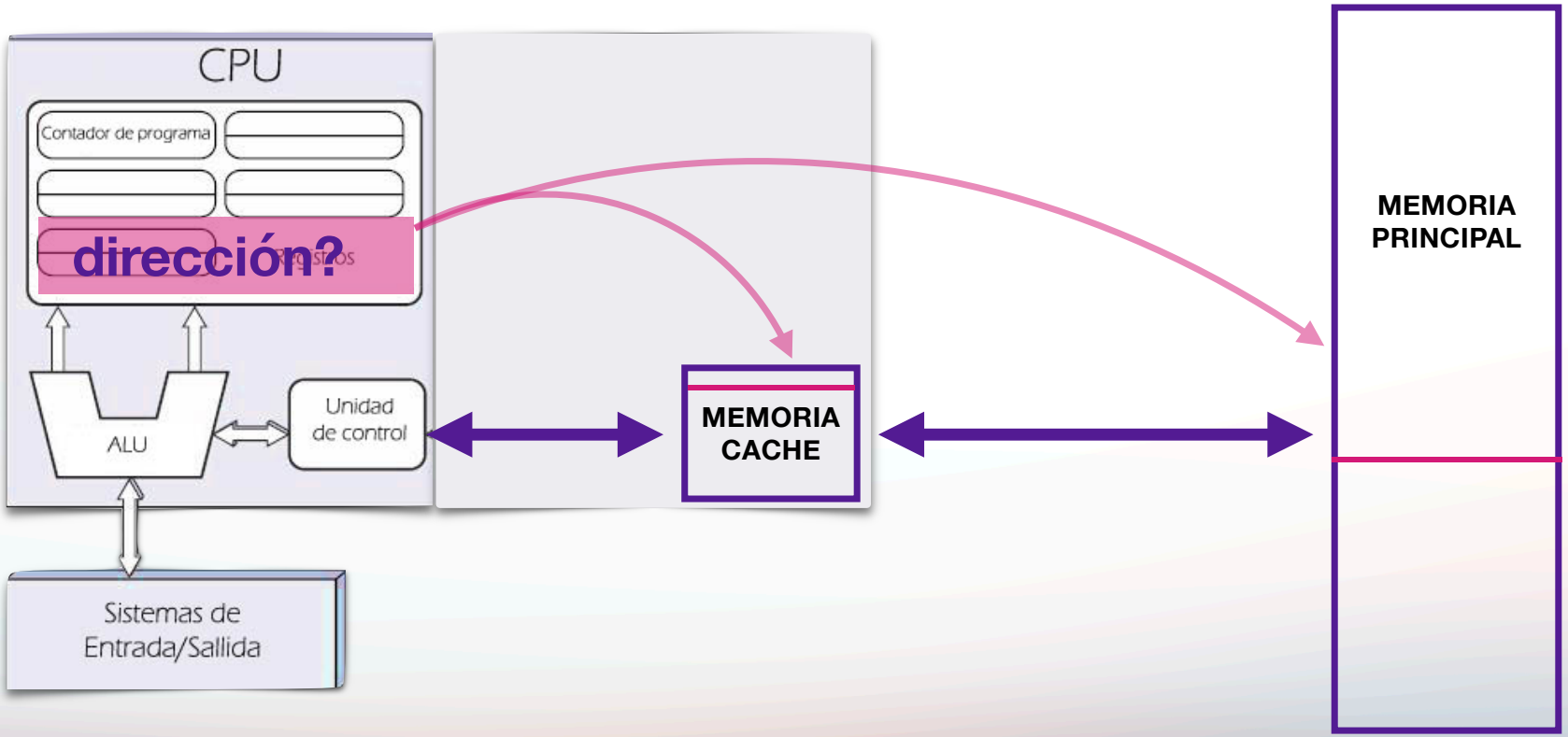
Memoria Cache - Idea

Cuando se necesita acceder a una posición se consulta a la memoria cache y principal, si se encuentra en la memoria cache (hit) se utiliza. Sino (miss), se espera de la memoria principal, NO sólo la información de la dirección sino también las que se encuentran próximas, así bajo el principio de localidad, el próximo acceso seguramente lo encontrará en la cache.



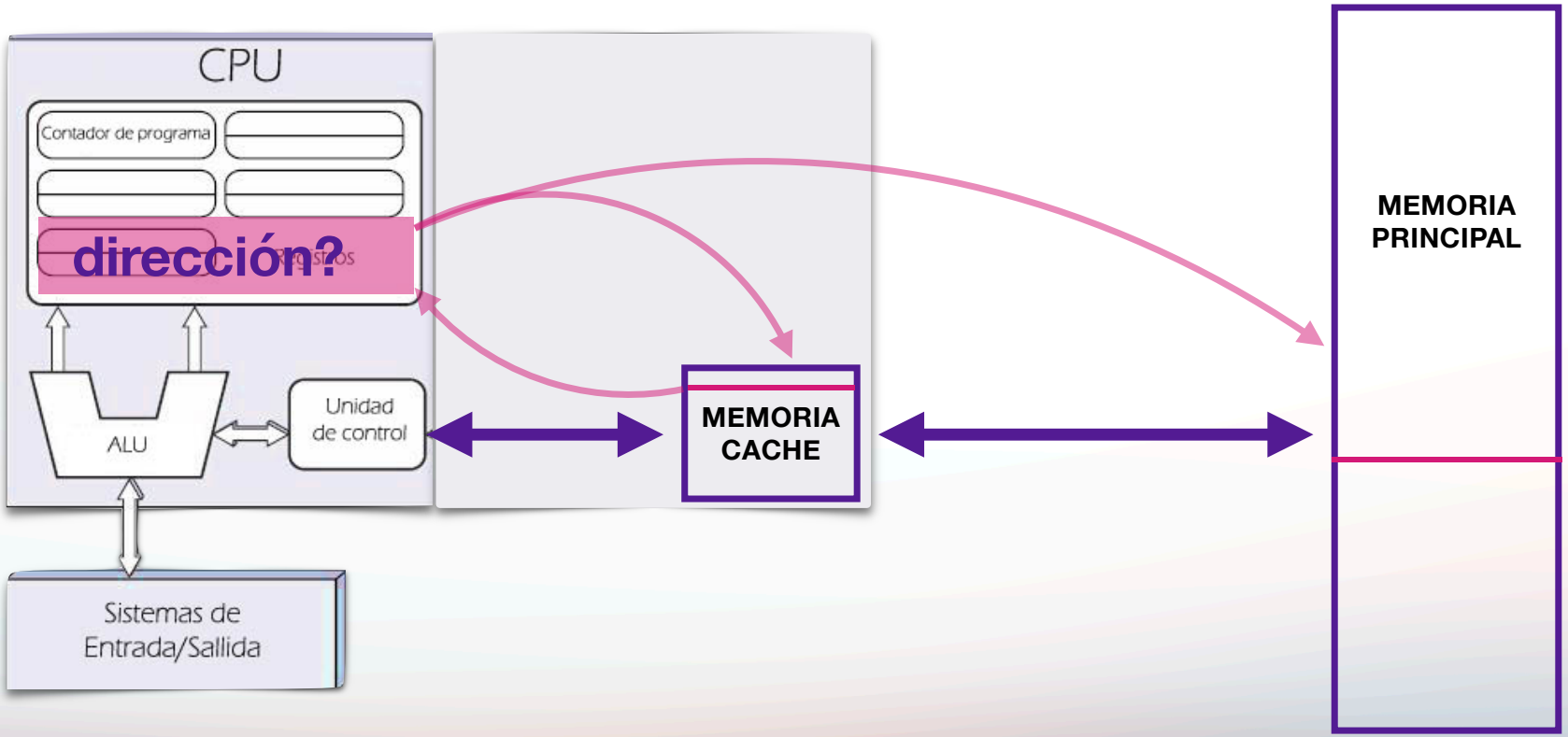
Memoria Cache - Idea

Cuando se necesita acceder a una posición se consulta a la memoria cache y principal, si se encuentra en la memoria cache (hit) se utiliza. Sino (miss), se espera de la memoria principal, NO sólo la información de la dirección sino también las que se encuentran próximas, así bajo el principio de localidad, el próximo acceso seguramente lo encontrará en la cache.



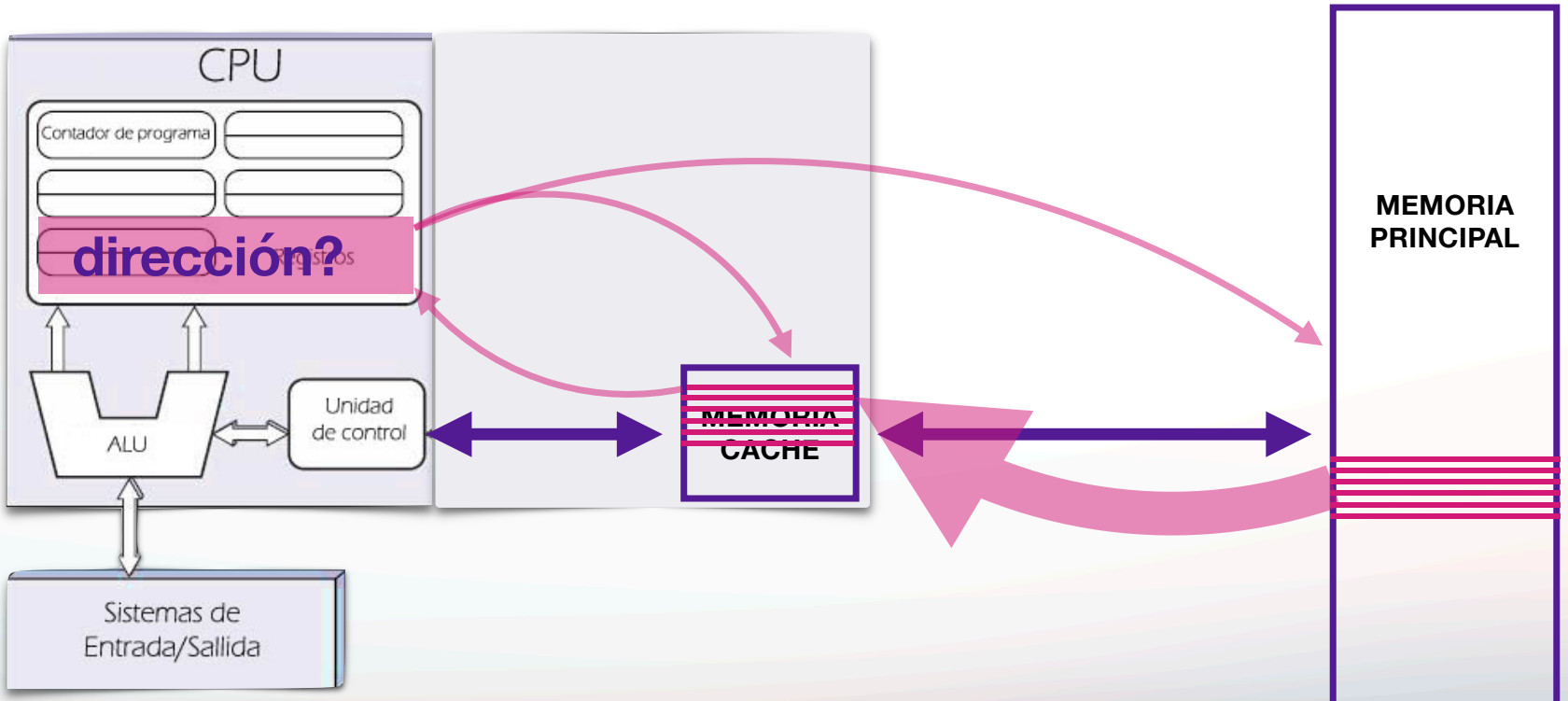
Memoria Cache - Idea

Cuando se necesita acceder a una posición se consulta a la memoria cache y principal, si se encuentra en la memoria cache (hit) se utiliza. Sino (miss), se espera de la memoria principal, NO sólo la información de la dirección sino también las que se encuentran próximas, así bajo el principio de localidad, el próximo acceso seguramente lo encontrará en la cache.



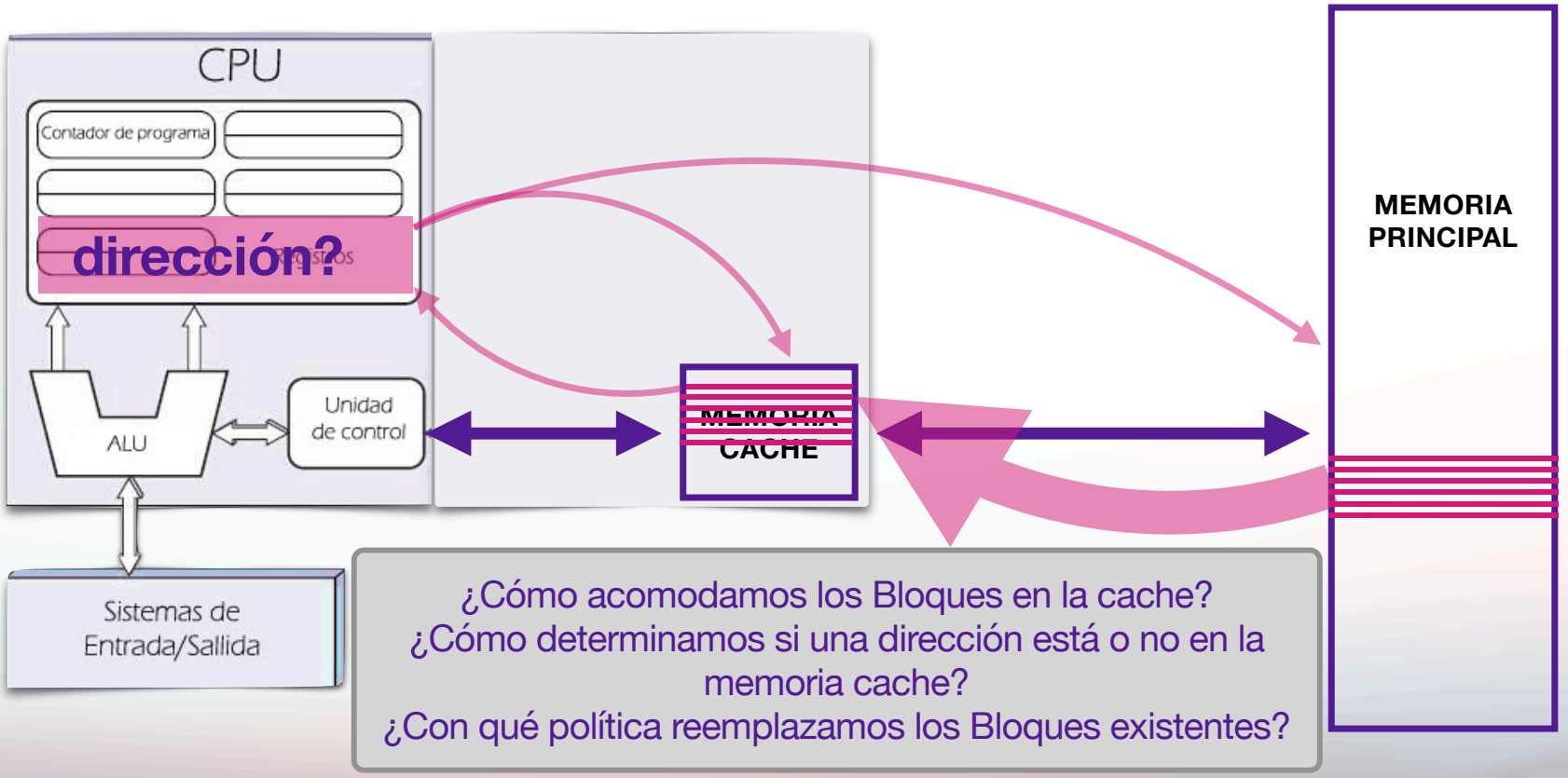
Memoria Cache - Idea

Cuando se necesita acceder a una posición se consulta a la memoria cache y principal, si se encuentra en la memoria cache (hit) se utiliza. Sino (miss), se espera de la memoria principal, NO sólo la información de la dirección sino también las que se encuentran próximas, así bajo el principio de localidad, el próximo acceso seguramente lo encontrará en la cache.



Memoria Cache - Idea

Cuando se necesita acceder a una posición se consulta a la memoria cache y principal, si se encuentra en la memoria cache (hit) se utiliza. Sino (miss), se espera de la memoria principal, NO sólo la información de la dirección sino también las que se encuentran próximas, así bajo el principio de localidad, el próximo acceso seguramente lo encontrará en la cache.



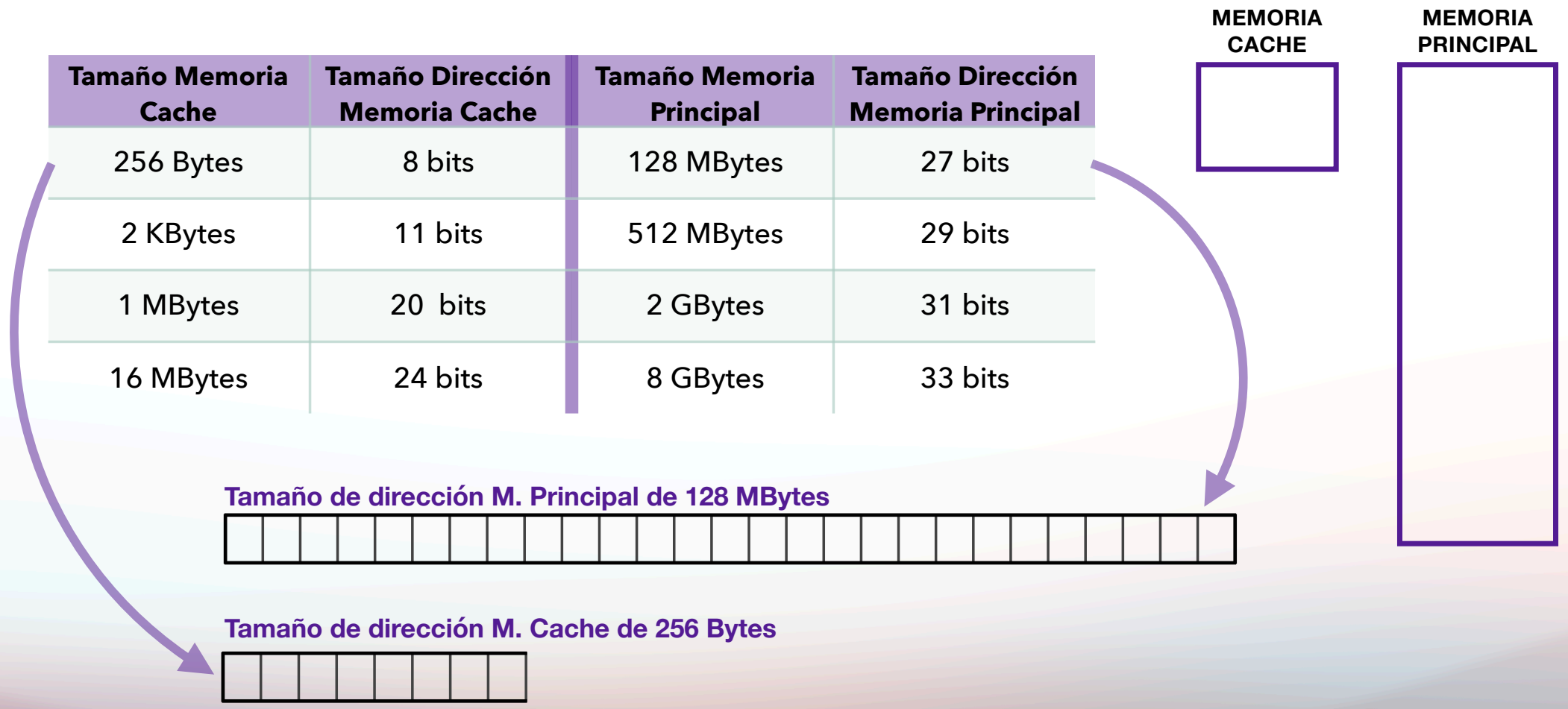
Memoria Cache - Terminología

- **Hit**: Los datos requeridos de la memoria residen en la memoria (niveles más altos).
- **Miss**: Los datos requeridos de la memoria NO residen en la memoria (niveles más altos).
- **Hit rate**: Porcentaje de Hits.
- **Miss rate**: Porcentaje de Miss.
- **Hit time**: Tiempo de acceso cuando los datos residen en la memoria.
- **Miss Penalty**: Tiempo adicional requerido para el acceso cuando los datos NO residen en la memoria.

Memoria Cache - sobre las direcciones

La primera consideración a tener en cuenta es el tamaño de las direcciones que depende el tamaño total de memoria (RAM y Cache). Algunos ejemplos:

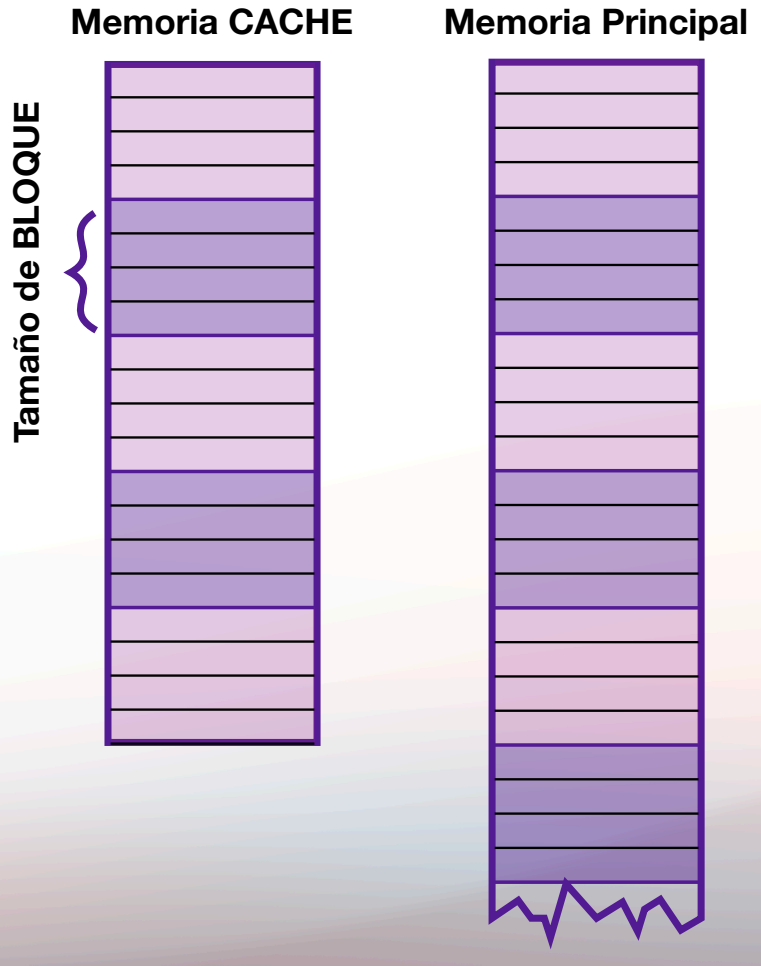
Tamaño Memoria Cache	Tamaño Dirección Memoria Cache	Tamaño Memoria Principal	Tamaño Dirección Memoria Principal
256 Bytes	8 bits	128 MBytes	27 bits
2 KBytes	11 bits	512 MBytes	29 bits
1 MBytes	20 bits	2 GBytes	31 bits
16 MBytes	24 bits	8 GBytes	33 bits



Memoria Cache - sobre los tamaños de bloques

Una decisión a tomar es el tamaño de los bloques. Es decir el contenido de cuántas direcciones, además de la requerida, que se cargan desde la memoria Principal hacia la cache en presencia de un miss.

- El tamaño de bloque es **fijo**. Por ejemplo 5 KBytes.
- Recordar que varios programas pueden estar ejecutándose en simultáneo y solicitando accesos a memoria (Cada uno de ellos con su propia “localía”)
- Bloques más grandes “aumentan” la probabilidad de hits, pero reducen la posibilidad de utilización compartida de la memoria cache (más reemplazos).
- Bloques más pequeño por el contrario maximiza en uso compartido por varios procesos/programas pero disminuye la tasa de aciertos



Memoria Cache - administración de bloques (correspondencia)

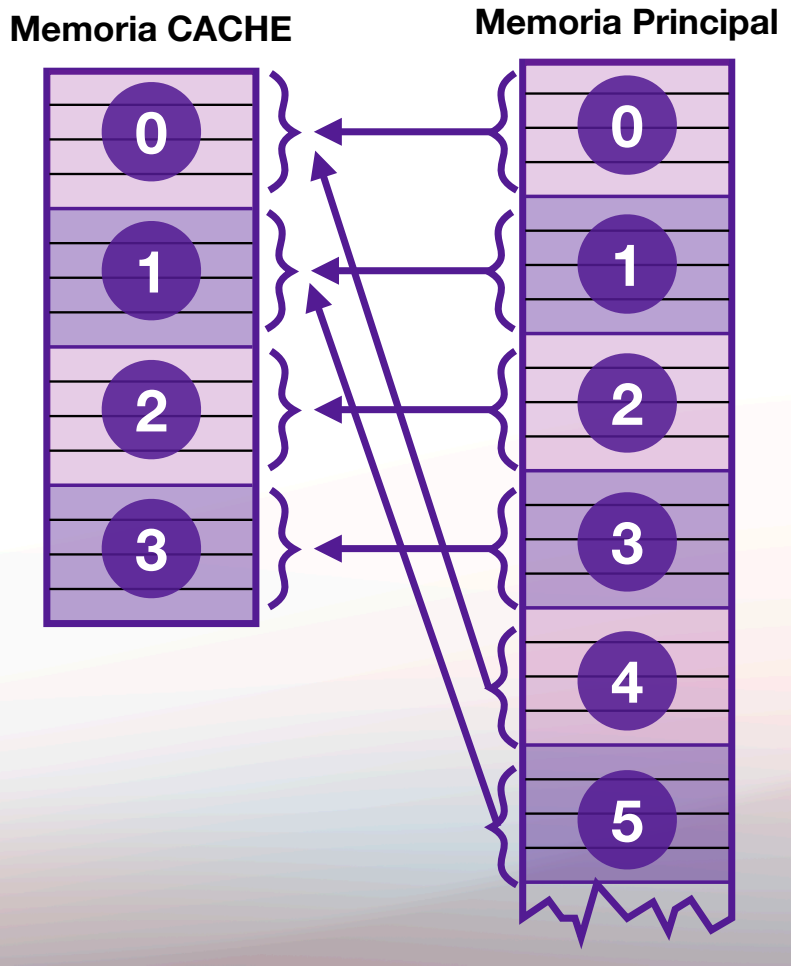
Una vez definido el tamaño de los bloques y dada la evidente diferencia en la cantidad de los mismos, existen diferentes formas de administrar los mismos, es decir, cómo asociamos (copiamos) bloques de la memoria Principal en la Cache:

- **Directa** (Direct Mapping): Cada bloque de la memoria Principal **SOLO** puede estar en **un lugar específico** en la memoria Cache.
- **Asociativa Completa** (Fully Associative Mapping): Cada bloque de la memoria Principal puede estar en **cualquier lugar** en la memoria Cache.
- **Asociativa por Conjuntos** (Set Associative Mapping): Cada bloque de la memoria Principal puede estar en **cualquier lugar de un conjunto** de la memoria Cache

Memoria Cache - correspondencia DIRECTA (Direct Mapping)

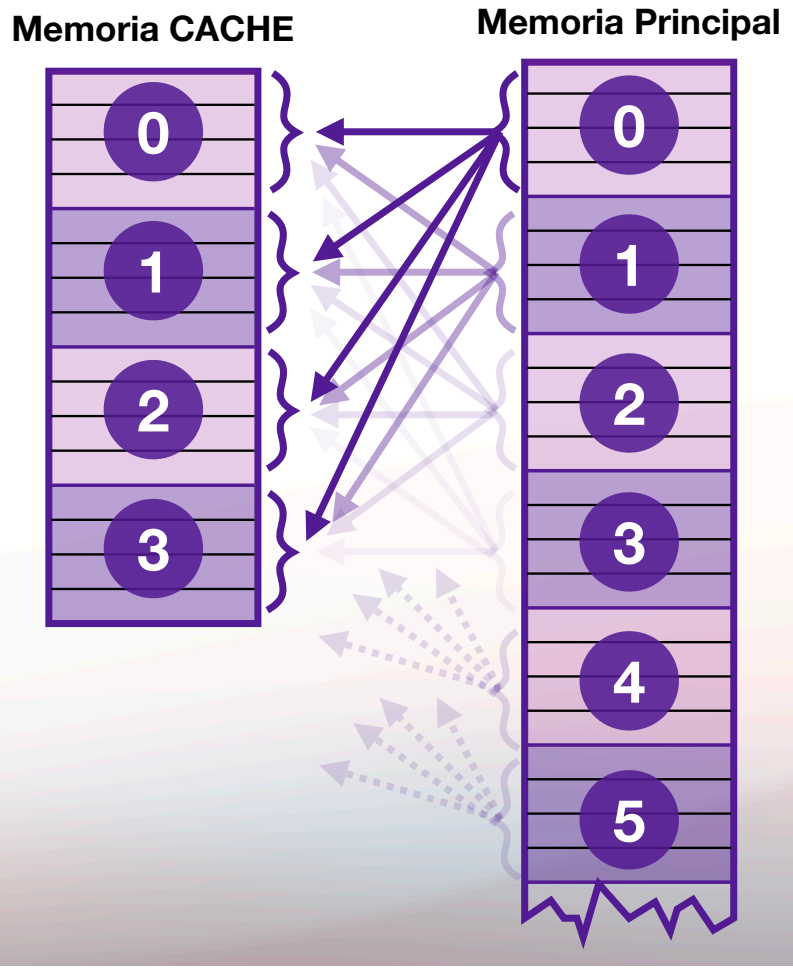
Cada bloque de la memoria Principal **SOLO** puede estar en **un lugar específico** en la memoria Cache.

- Es la más eficiente de la organizaciones en cuanto a costo de implementación ya que dada una dirección de memoria principal sólo puede estar en un Bloque de la memoria Cache. No se necesita “buscar” en cada posible bloque.
- Como contraparte es la que menos maximiza la utilización de memoria Cache. Por ejemplo, en este simple escenario, si los se estuviera utilizando en simultáneo direcciones de la memoria Principal correspondientes a los bloques 1 y 5 , se producirían continuos reemplazos, pese a que el resto de la memoria Cache pudiera estar sin utilizarse.



Memoria Cache - correspondencia ASOCIATIVA COMPLETA (Fully Associative Mapping)

Cada bloque de la memoria Principal puede estar en **cualquier lugar** en la memoria Cache.

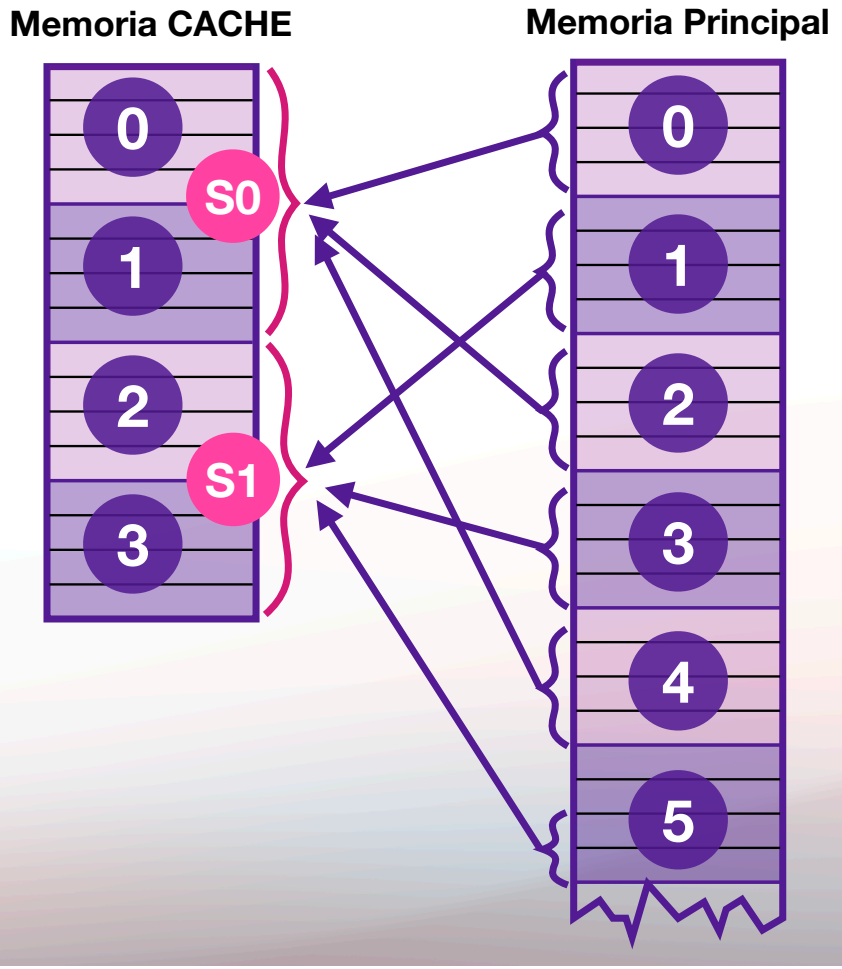


- En contraposición con la Directa, la correspondencia asociativa completa, maximiza la utilización de la memoria permitiendo que los bloques de la memoria principal puedan residir en cualquier lugar de la memoria cache. Sólo comenzarán los reemplazos cuando la memoria cache esté completamente utilizada.
- Como contraparte su implementación es más costosa (prohibitiva) ya que se debe implementar una búsqueda (hardware) paralela en toda la memoria cache.

Memoria Cache - correspondencia ASOCIATIVA por Conjuntos (Set Associative Mapping)

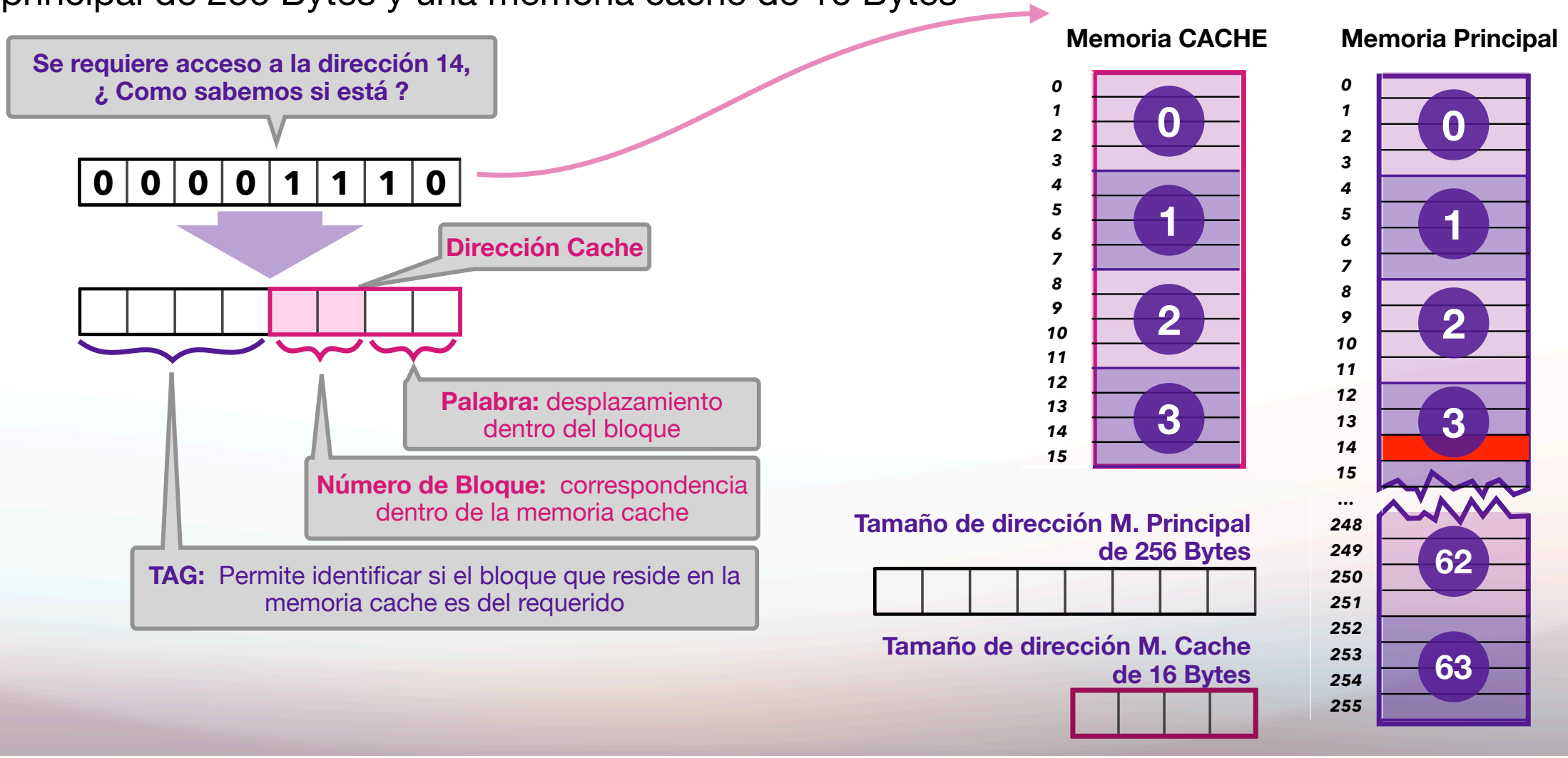
Cada bloque de la memoria Principal puede estar en **cualquier lugar de un conjunto** de la memoria Cache

- Esta es una correspondencia que captura lo mejor de las anteriores. Por un lado mantiene el costo de implementación viable mientras promueve la maximización de la utilización de la memoria cache
- Ahora en el escenario planteado para correspondencia directa, si los bloques 1 y 5 se utilizaran de manera simultánea, ambos podrían residir en el mismo conjunto **S1**.
- Cabe notar que dependiendo la cantidad ***n*** de bloques por conjunto (***n - asociativa***) nos movemos hacia una correspondencia más directa o más completa asociativa. Por ejemplo 1 asociativa es igual que correspondencia directa.



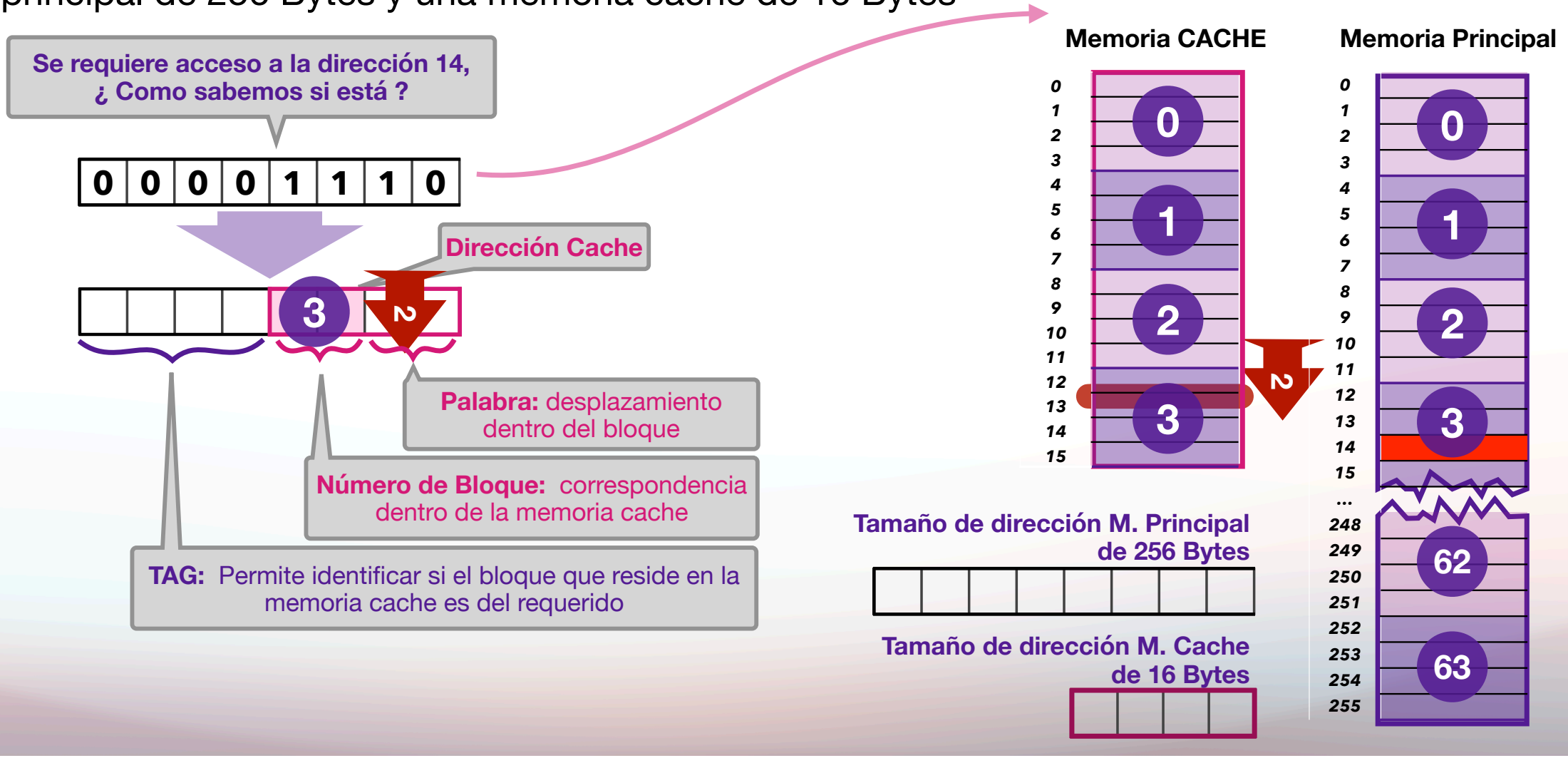
Memoria Cache - Cómo para determinar si la dirección requerida está en Cache (Directo)

Para comprender el proceso asumamos un simple escenario con una memoria principal de 256 Bytes y una memoria cache de 16 Bytes



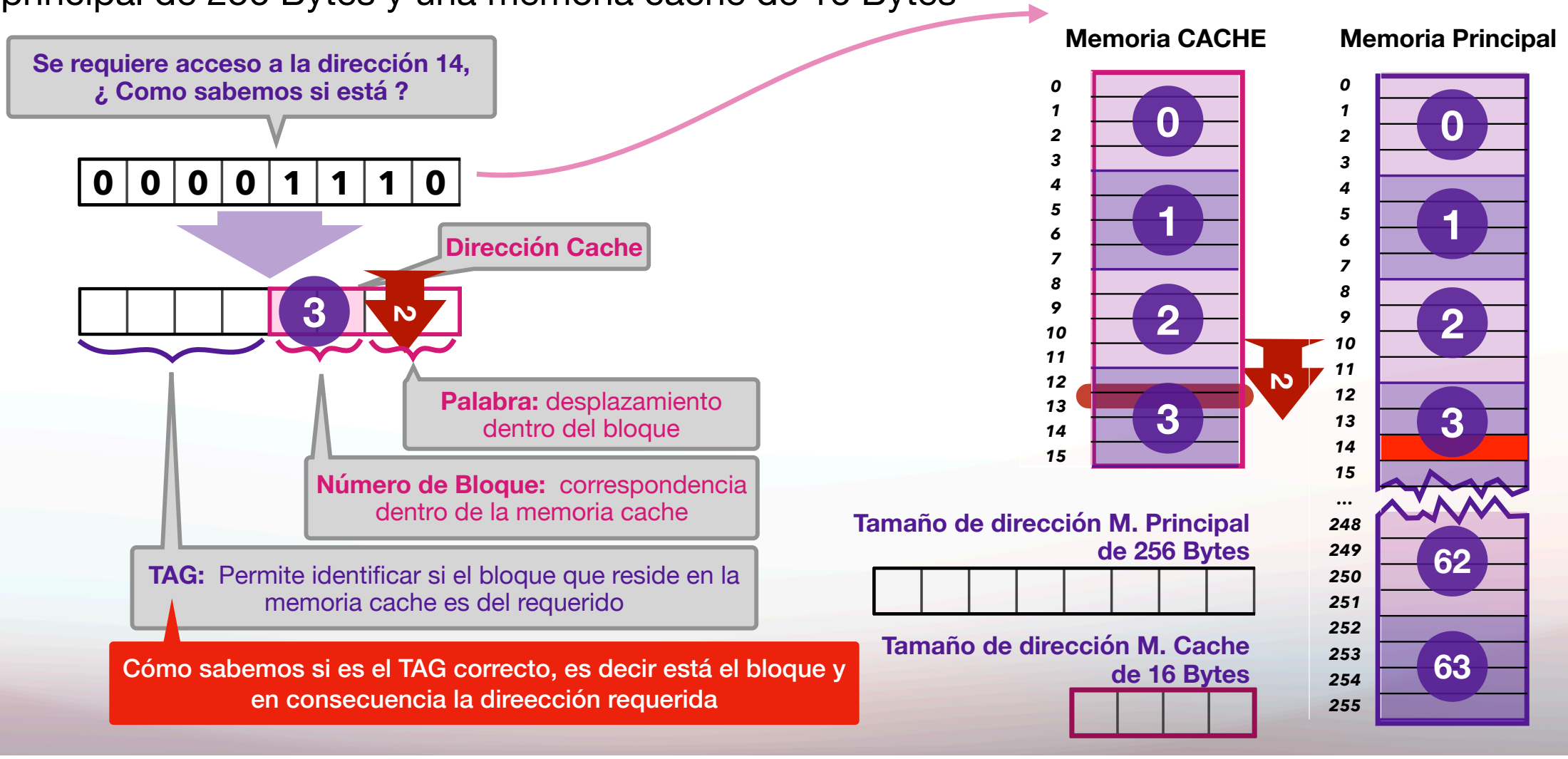
Memoria Cache - Cómo para determinar si la dirección requerida está en Cache (Directo)

Para comprender el proceso asumamos un simple escenario con una memoria principal de 256 Bytes y una memoria cache de 16 Bytes



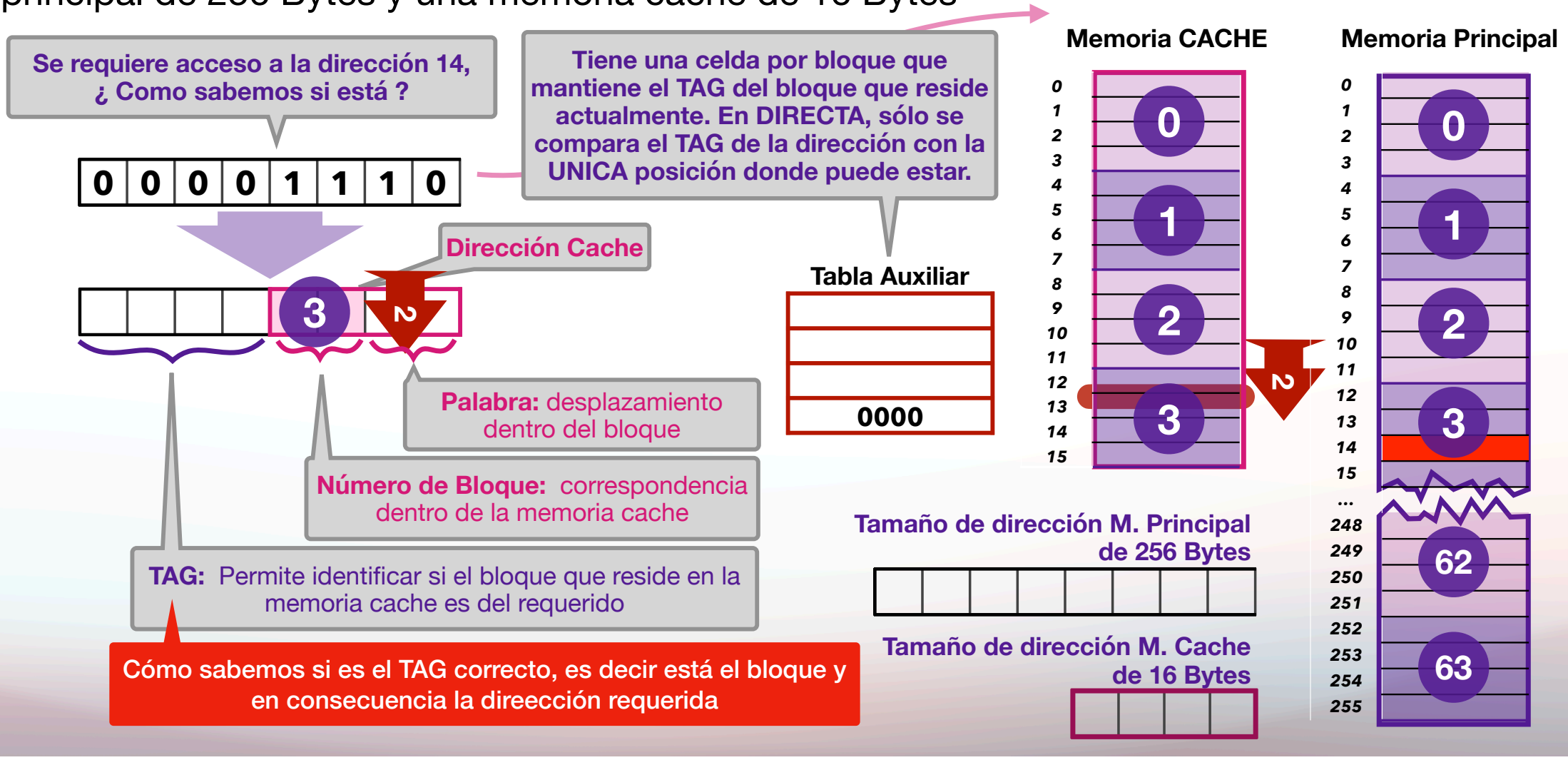
Memoria Cache - Cómo para determinar si la dirección requerida está en Cache (Directo)

Para comprender el proceso asumamos un simple escenario con una memoria principal de 256 Bytes y una memoria cache de 16 Bytes



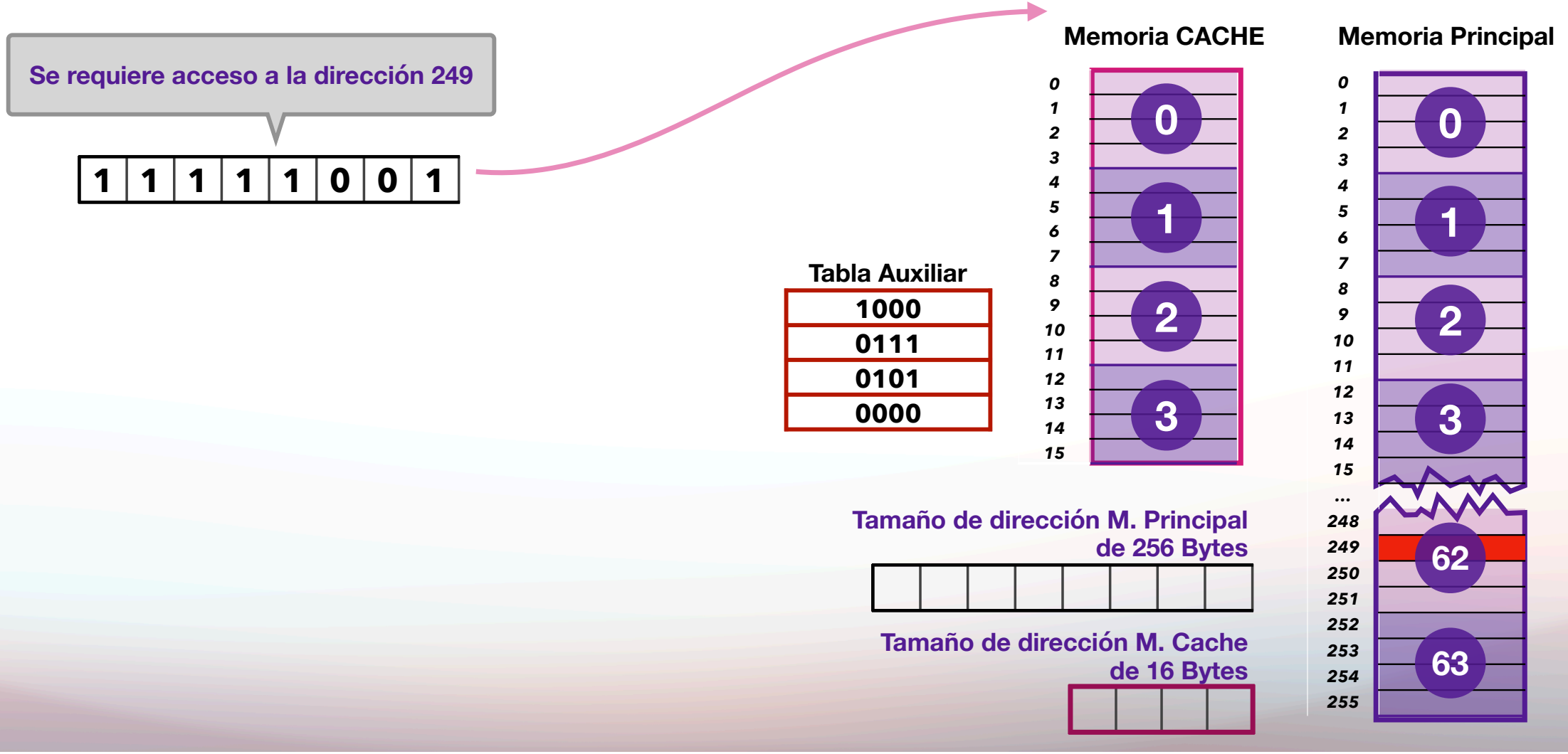
Memoria Cache - Cómo para determinar si la dirección requerida está en Cache (Directo)

Para comprender el proceso asumamos un simple escenario con una memoria principal de 256 Bytes y una memoria cache de 16 Bytes



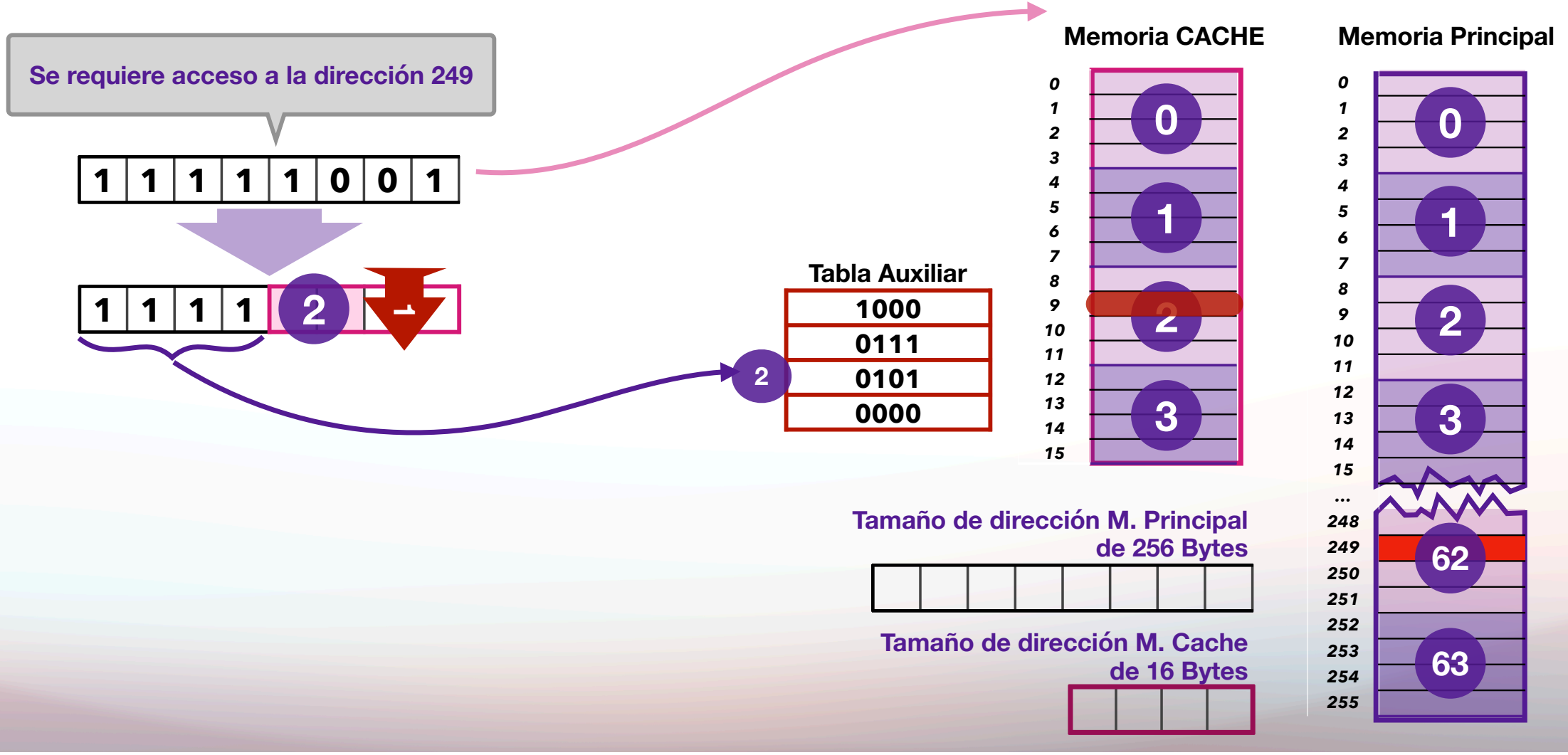
Memoria Cache - Cómo para determinar si la dirección requerida está en Cache (Directo)

Ejemplo con otra solicitud con un estado actual diferente



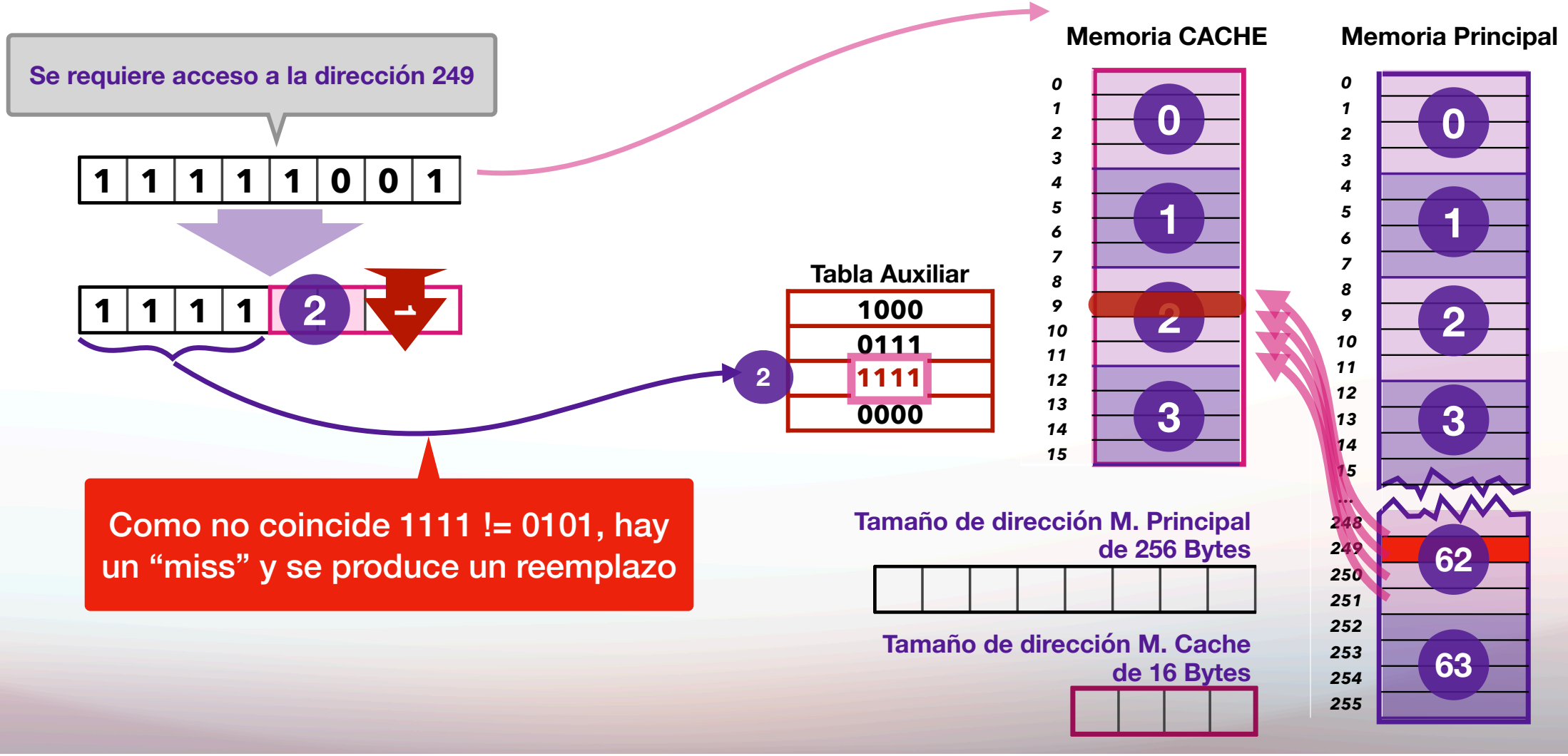
Memoria Cache - Cómo para determinar si la dirección requerida está en Cache (Directo)

Ejemplo con otra solicitud con un estado actual diferente



Memoria Cache - Cómo para determinar si la dirección requerida está en Cache (Directo)

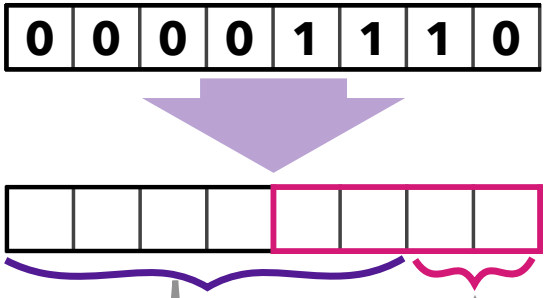
Ejemplo con otra solicitud con un estado actual diferente



Memoria Cache - Cómo para determinar si la dirección requerida está en Cache (Asociativa Completa)

Para comprender el proceso asumamos un simple escenario con una memoria principal de 256 Bytes y una memoria cache de 16 Bytes

Se requiere acceso a la dirección 14, ¿ Como sabemos si está ?



Palabra: desplazamiento dentro del bloque

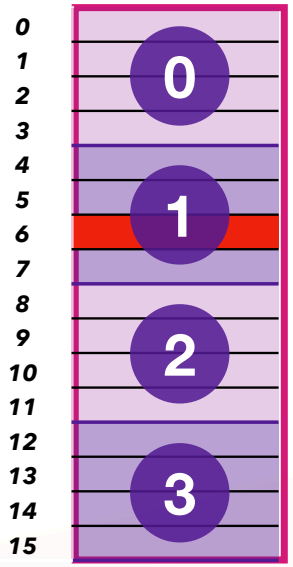
TAG: Dado que los bloque pueden residir en cualquier posición para poder determinar su posición debemos mantener y BUSCAR en la tabla auxiliar por un TAG más grande que incluye los bits que no sean de la **palabra.**

Por ejemplo una dirección del bloque de memoria principal **2** y **3** podrían residir en el bloque **1** de la cache y ambos comparten el prefijo 0000.

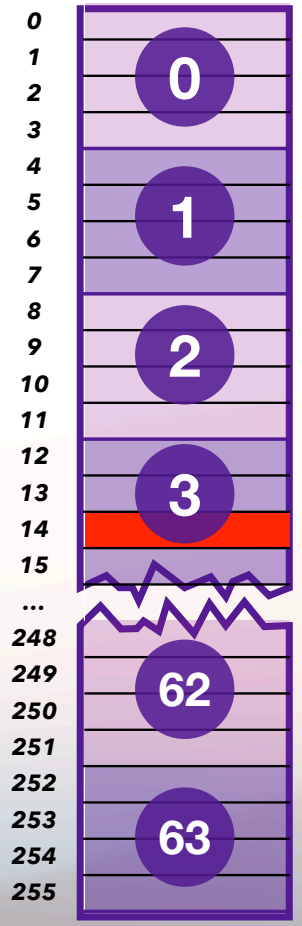
Tabla Auxiliar

000011

Memoria CACHE



Memoria Principal

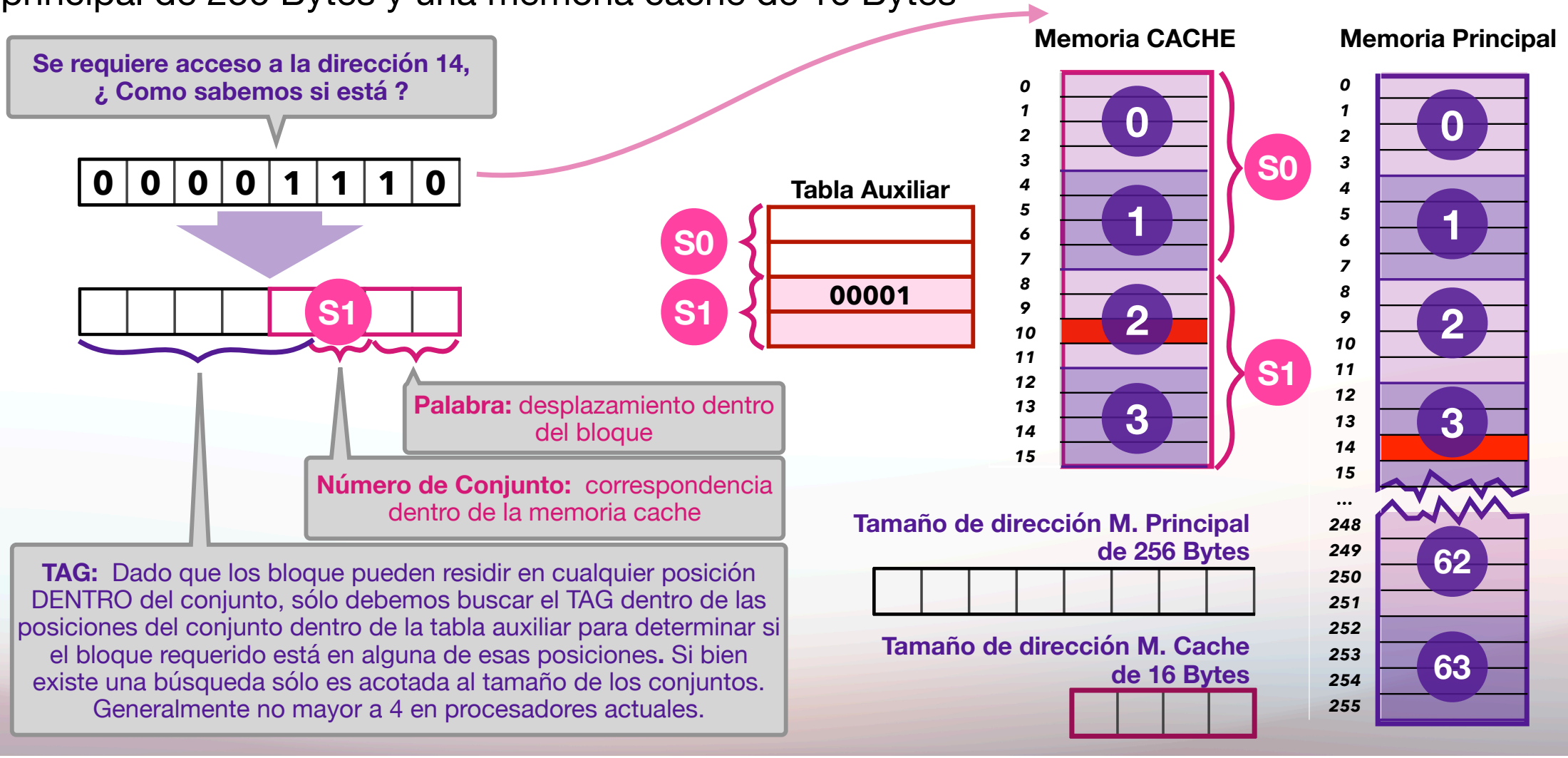


Tamaño de dirección M. Principal de 256 Bytes

Tamaño de dirección M. Cache de 16 Bytes

Memoria Cache - Cómo para determinar si la dirección requerida está en Cache (Asociativa por Conjuntos)

Para comprender el proceso asumamos un simple escenario con una memoria principal de 256 Bytes y una memoria cache de 16 Bytes

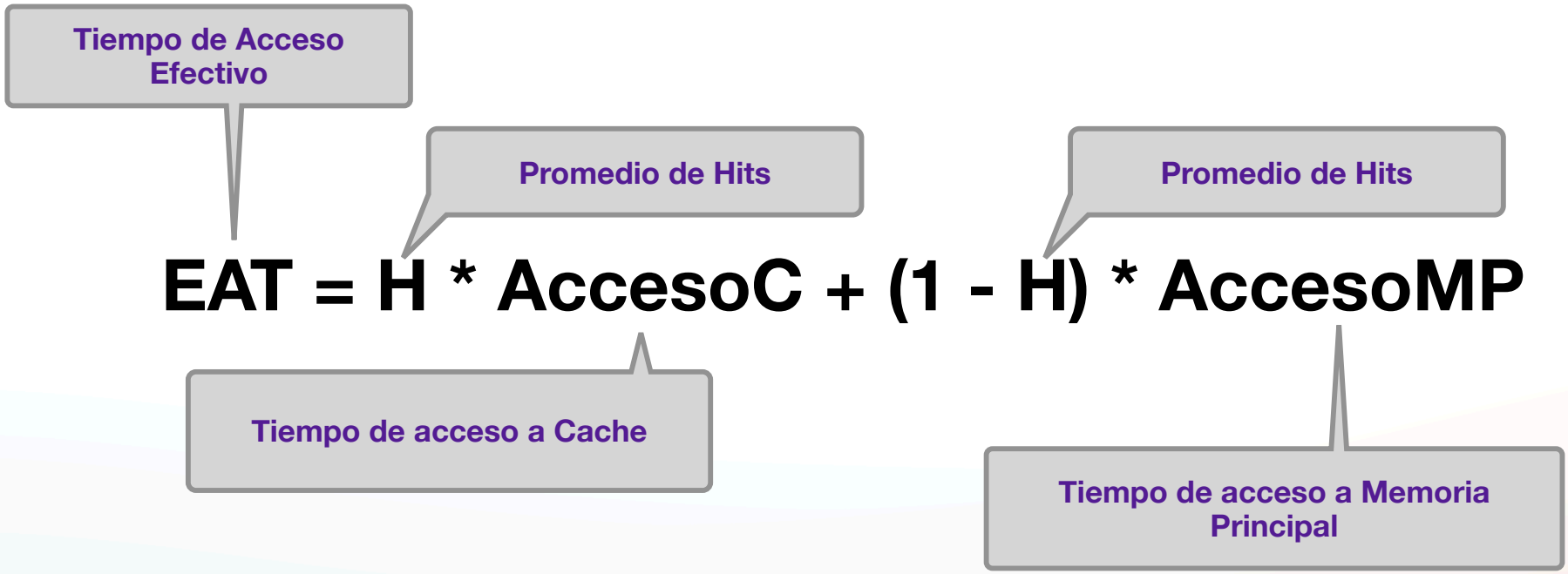


Memoria Cache - Tipos de reemplazo

Cuando un Bloque de la memoria cache necesita ser cargado desde la memoria por un miss y si la cache está llena (algo que debe suceder) debemos elegir un Bloque a reemplazar. Si la correspondencia es Directa, la decisión es trivial, sino :

- **LRU** (Least Recently Used): el más antiguo debe ser reemplazado.
- **FIFO** (First In First Out): comportamiento de cola.
- **RADOM**: se elige uno aleatoriamente.

Memoria Cache - Tiempo de Acceso Efectivo y porcentaje de hits



Memoria Cache - Políticas de escritura

Cuando el contenido de una dirección de memoria (que reside en la cache) es modificado, debemos tomar una política de actualización de la memoria principal para que la información quede consistente:

- **Write through:** cuando se modifica el valor en la memoria cache, se propaga y se modifica en la memoria principal.
- **Write back:** sólo se modifica en la memoria caché hasta que el Bloque se marcado como víctima para ser reemplazado, en ese momento se copia todo el Bloque a la memoria.

Organización del Procesador

Memoria Virtual

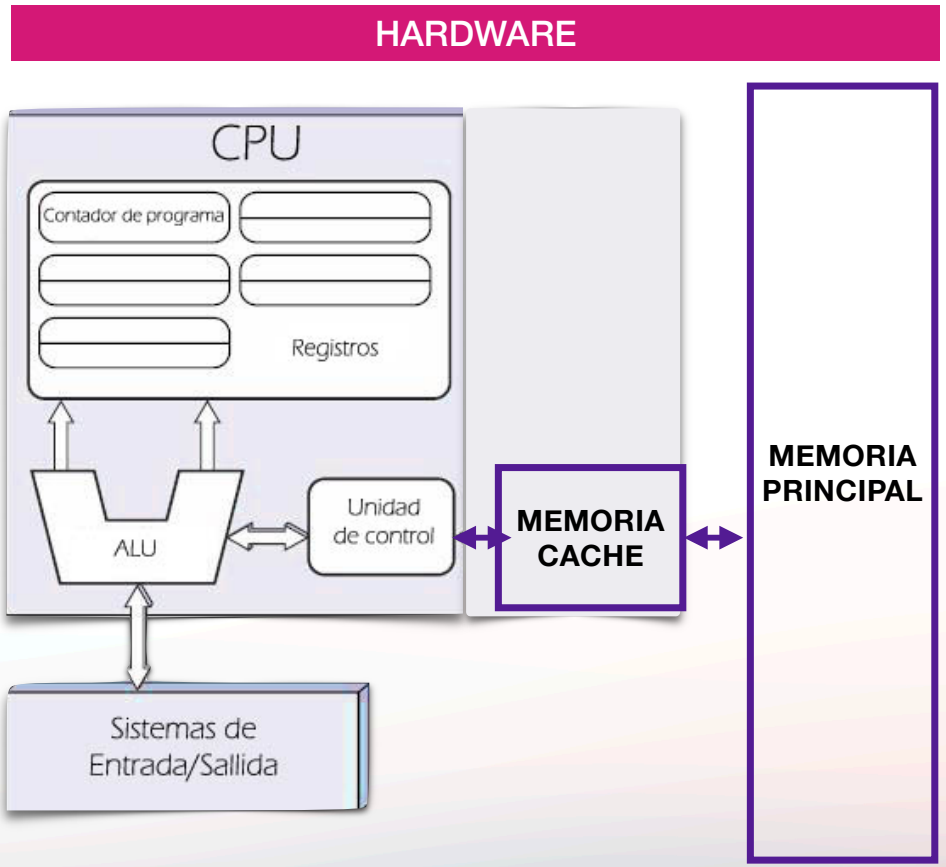
Departamento de Computación - UNRC

Memoria Virtual

La técnica de memoria virtual tiene como objetivo maximizar el uso de la memoria, distribuyendo su asignación de manera controlada a los procesos que están ejecutándose.

La virtualización de la memoria está **a cargo del Sistema Operativo** (Software).

Existen diferentes modos que general mente se utilizan de manera combinada. Los usualmente utilizados son Paginación y Segmentación

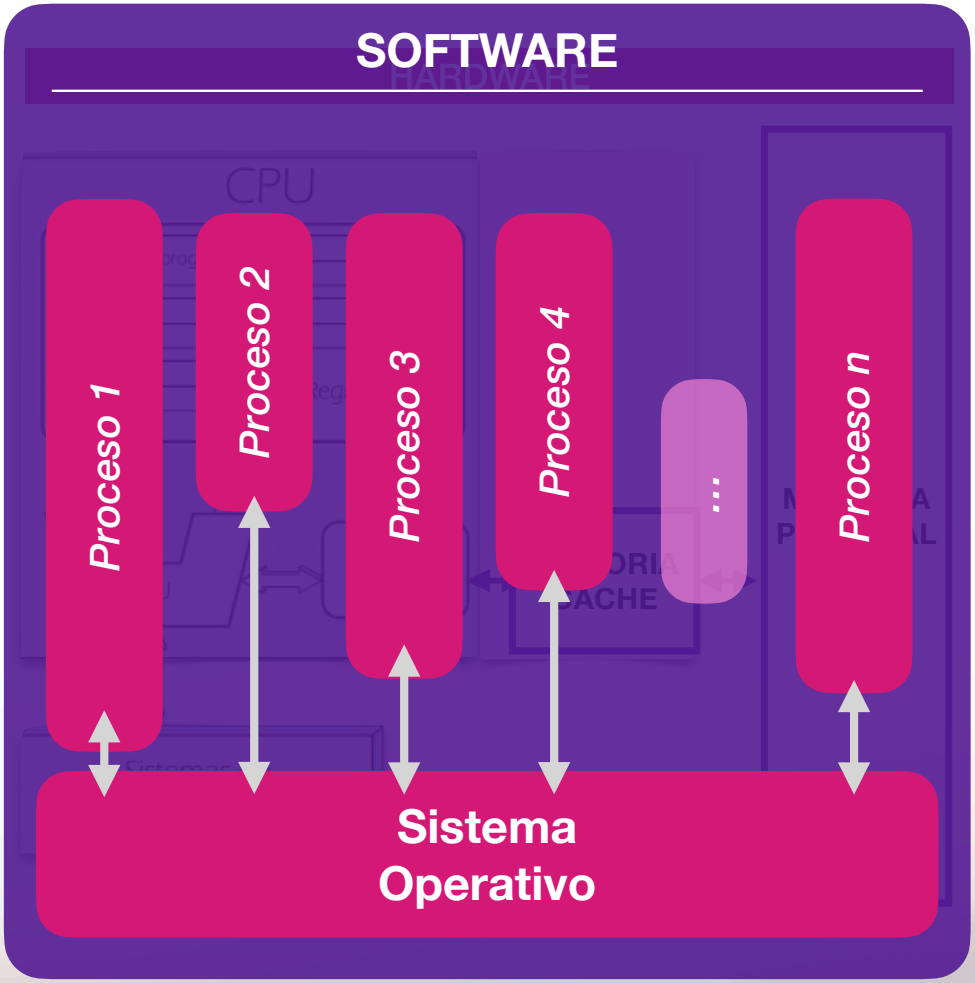


Memoria Virtual

La técnica de memoria virtual tiene como objetivo maximizar el uso de la memoria, distribuyendo su asignación de manera controlada a los procesos que están ejecutándose.

La virtualización de la memoria está **a cargo del Sistema Operativo** (Software).

Existen diferentes modos que general mente se utilizan de manera combinada. Los usualmente utilizados son Paginación y Segmentación

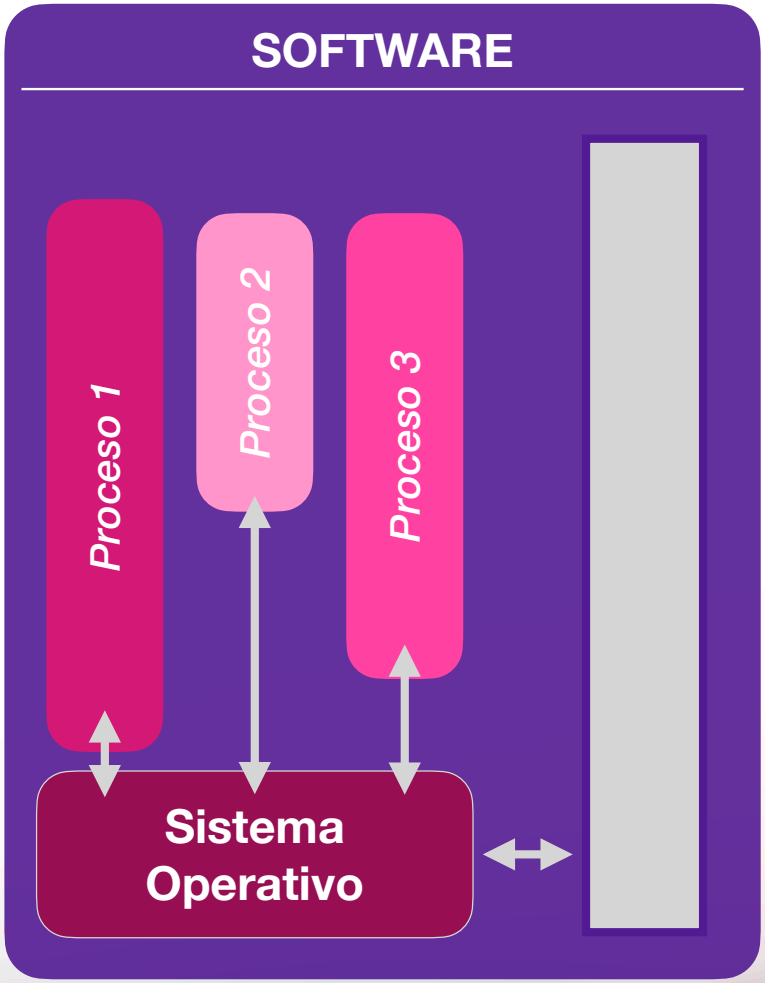


Memoria Virtual - Idea conceptual

Dado que cada proceso NO necesita necesariamente TODO el espacio de memoria al mismo tiempo, la idea (transparente para el proceso) es asignarle memoria en la medida que éstos procesos la van accediendo.

Por ello se fracciona el uso de la memoria en **Páginas** y/o **Segmentos** que le permite al Sistema Operativo ir administrando y distribuyendo la memoria disponible a los procesos en ejecución, en la medida que la van accediendo.

- **Paginado**: fracciona la memoria en **páginas del mismo tamaño**.
- **Segmentación**: fracciona la memoria en **segmentos de diferentes tamaños** dependiendo de la necesidad de cada proceso.

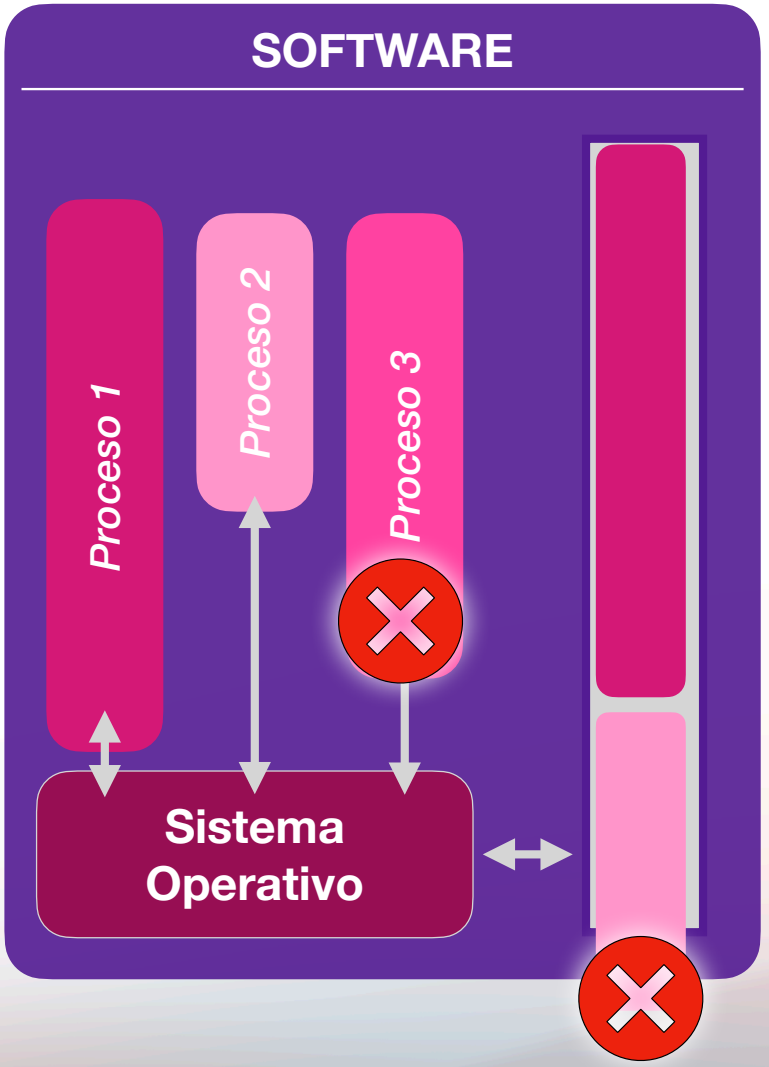


Memoria Virtual - Idea conceptual

Dado que cada proceso NO necesita necesariamente TODO el espacio de memoria al mismo tiempo, la idea (transparente para el proceso) es asignarle memoria en la medida que éstos procesos la van accediendo.

Por ello se fracciona el uso de la memoria en **Páginas** y/o **Segmentos** que le permite al Sistema Operativo ir administrando y distribuyendo la memoria disponible a los procesos en ejecución, en la medida que la van accediendo.

- **Paginado**: fracciona la memoria en **páginas del mismo tamaño**.
- **Segmentación**: fracciona la memoria en **segmentos de diferentes tamaños** dependiendo de la necesidad de cada proceso.

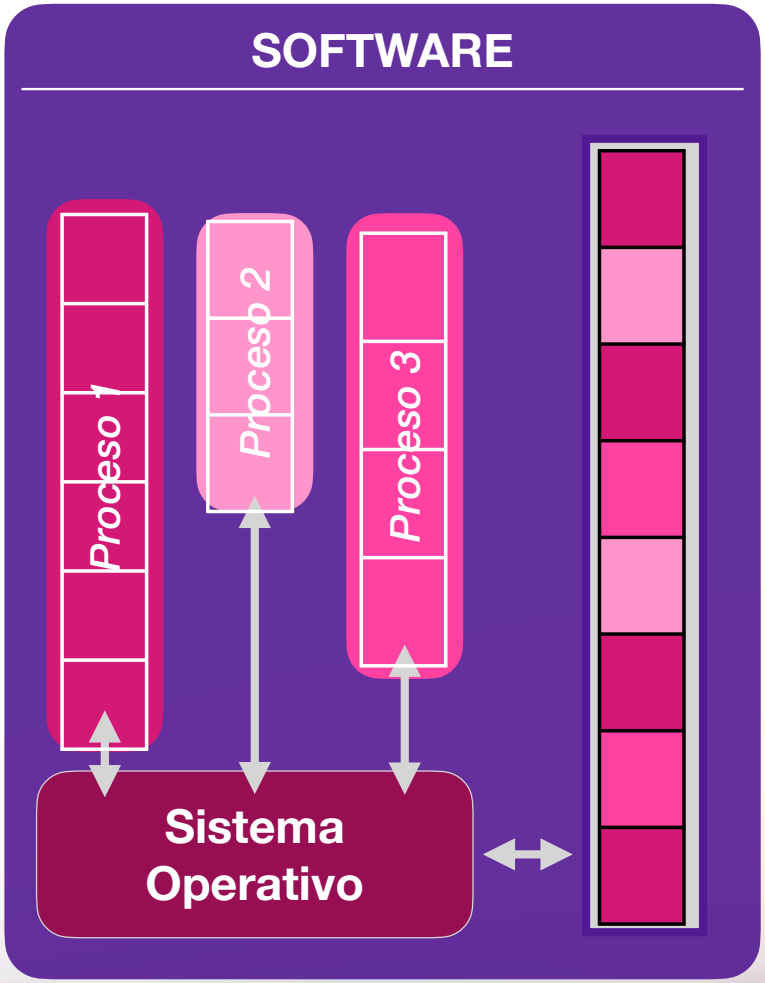


Memoria Virtual - Idea conceptual

Dado que cada proceso NO necesita necesariamente TODO el espacio de memoria al mismo tiempo, la idea (transparente para el proceso) es asignarle memoria en la medida que éstos procesos la van accediendo.

Por ello se fracciona el uso de la memoria en **Páginas** y/o **Segmentos** que le permite al Sistema Operativo ir administrando y distribuyendo la memoria disponible a los procesos en ejecución, en la medida que la van accediendo.

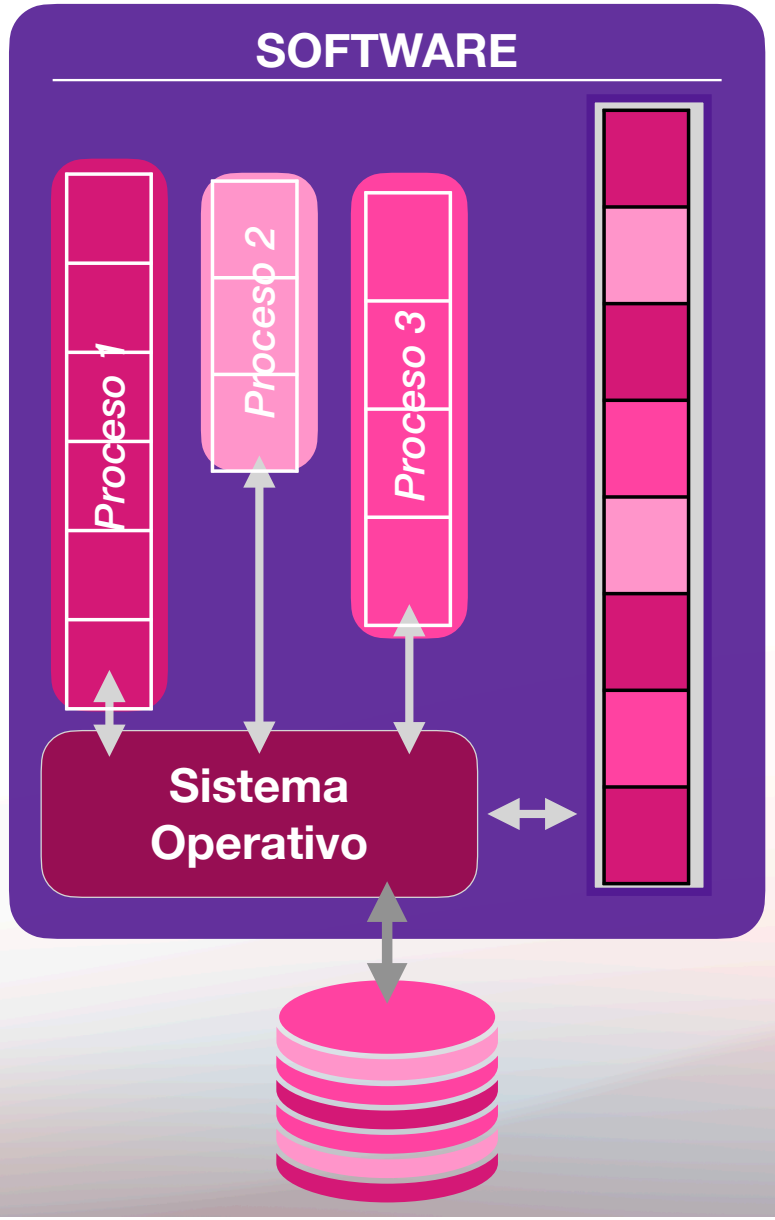
- **Paginado**: fracciona la memoria en **páginas del mismo tamaño**.
- **Segmentación**: fracciona la memoria en **segmentos de diferentes tamaños** dependiendo de la necesidad de cada proceso.



Memoria Virtual - Idea conceptual

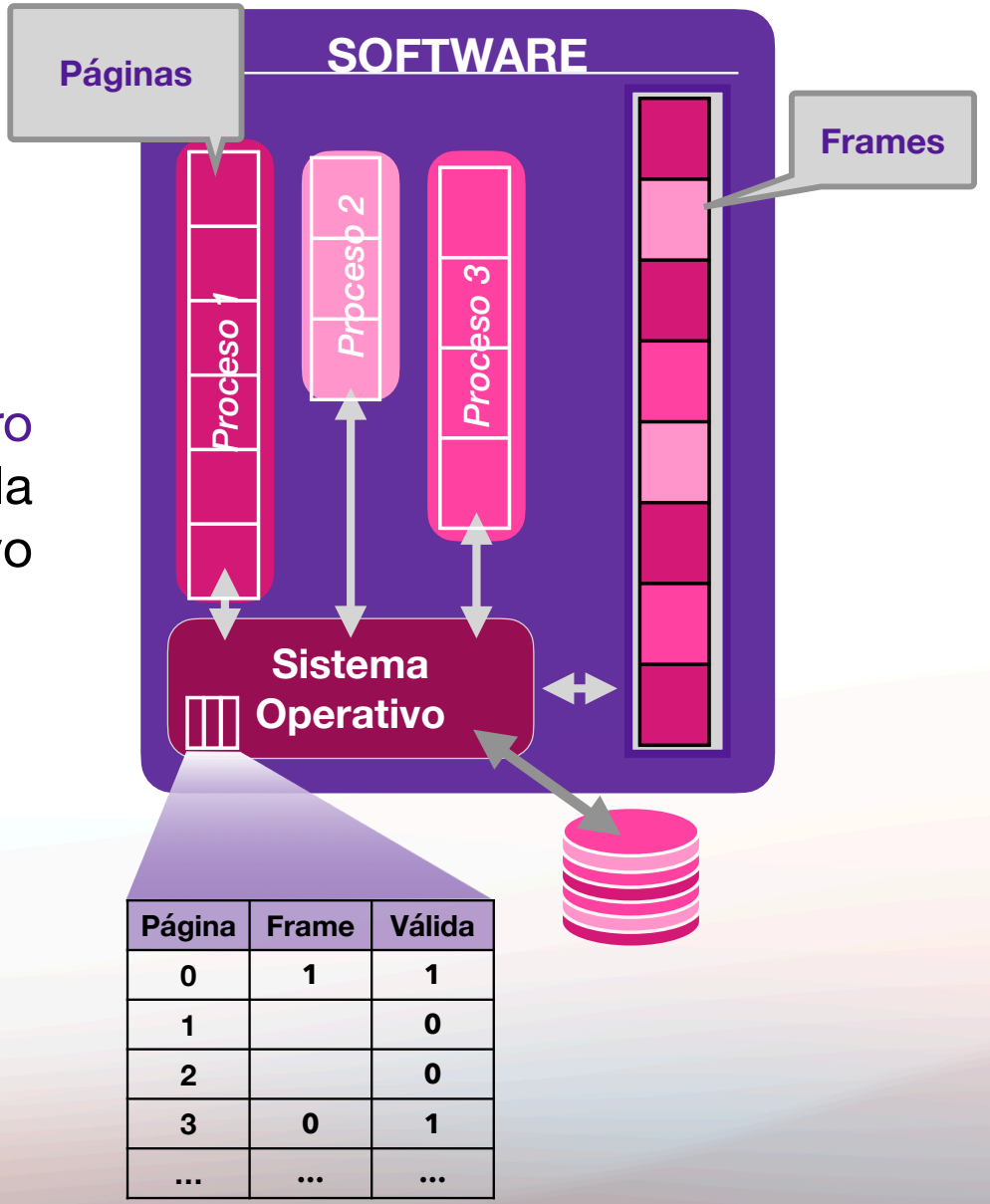
Como es un proceso administrado por el Sistema Operativo. Además de administrar el espacio físico de la memoria, se puede virtualizar (simular) más capacidad de memoria que la física, almacenando la información (contenido de la memoria) temporalmente en espacios de almacenamiento más grades, como por ejemplo, en los discos de almacenamiento HDD, SDD, etc.

Cuando la memoria física se completa, se comienza a utilizar otras unidades de almacenamiento para guardar temporalmente espacios de memoria. La desventaja es que la performance de la ejecución disminuye notablemente en la medida que crece el uso de estos espacios secundarios. Cuando la demanda es extrema se denomina **hiperpaginación** o **trashing**, donde el sistema operativo pasa más tiempo intercambiando páginas entre la memoria y el disco que ejecutando procesos.



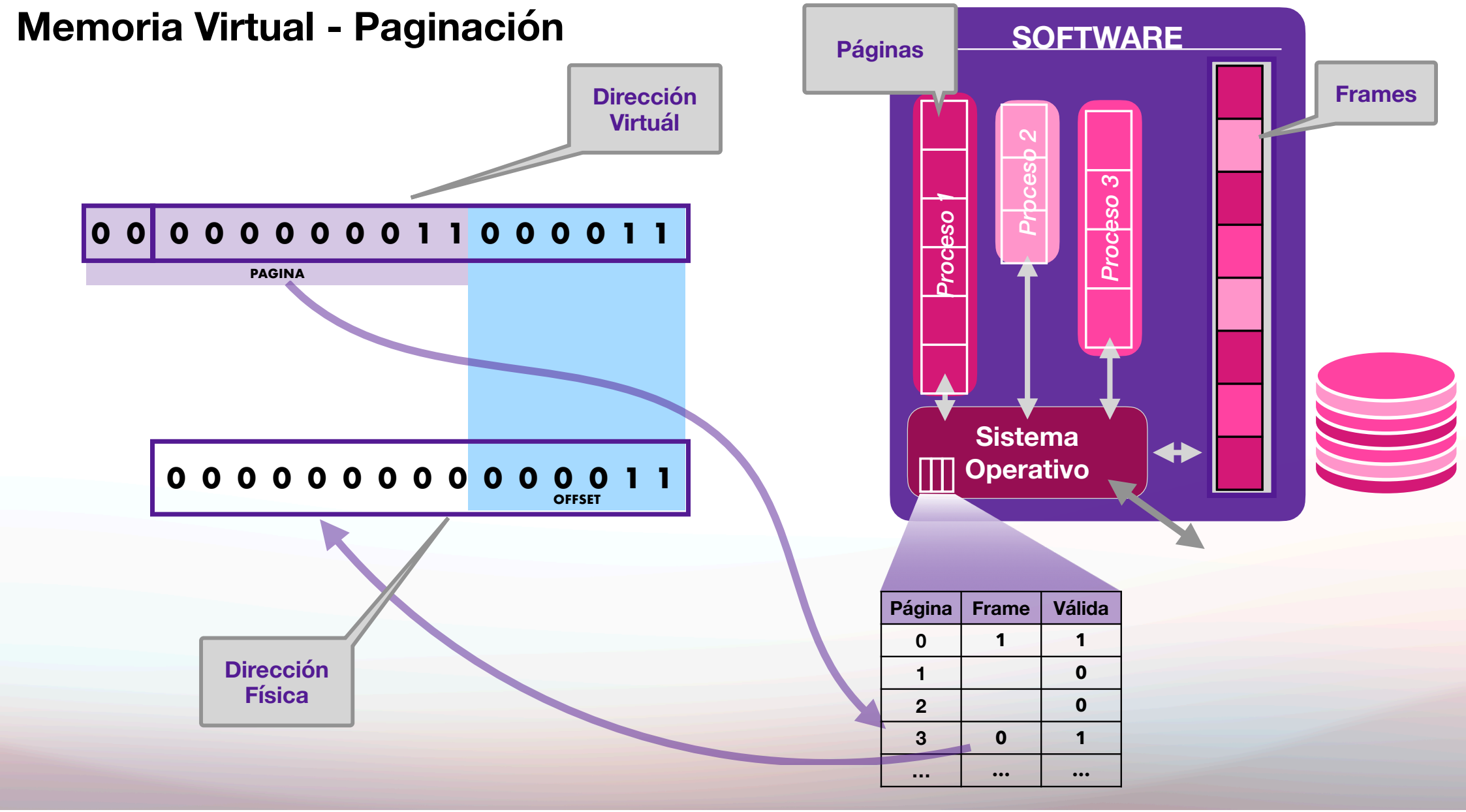
Memoria Virtual - Paginación

Para mantener la correspondencia entre el número de página asignado a un proceso y su lugar en la memoria (número de frame), el sistema operativo mantiene una **tabla de páginas.**



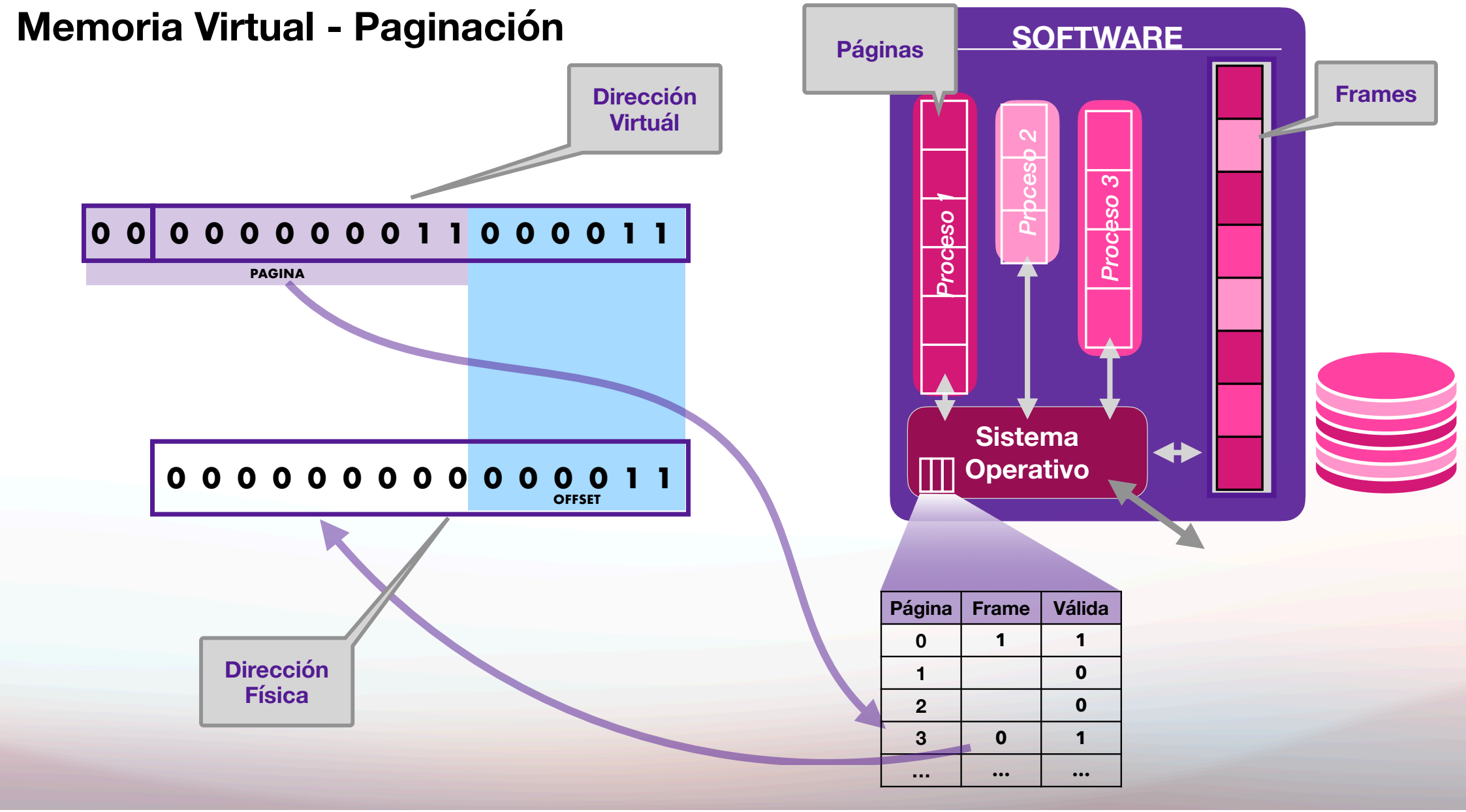
Página	Frame	Válida
0	1	1
1		0
2		0
3	0	1
...	...	...

Memoria Virtual - Paginación



Página	Frame	Válida
0	1	1
1		0
2		0
3	0	1
...	...	...

Memoria Virtual - Paginación



Página	Frame	Válida
0	1	1
1		0
2		0
3	0	1
...	...	...

Memoria Virtual - Paginación

MEMORIA PROGRAMA

Página	Dirección	Valor
0	0000	A
	0001	B
1	0010	C
	0011	D
2	0100	E
	0101	F
3	0110	G
	0111	H
4	1000	I
	1001	J
5	1010	K
	1011	L
6	1100	M
	1101	N
7	1110	O
	1111	P

MEMORIA FÍSICA

FRAME	Dirección	Valor
0	000	C
	001	D
1	010	G
	011	H
2	100	A
	101	B
3	110	K
	111	L

TABLA DE PÁGINAS

Página	FRAME	Valid
0	2	1
1	0	1
2	-	0
3	1	1
4	-	0
5	3	1
6	-	0
7	-	0

DIRECCIÓN VIRTUAL

4 BITS

K | 1 0 1 0

PAGINA | OFFSET

DIRECCIÓN FÍSICA

3 BITS

K | 1 1 0

PAGINA | OFFSET

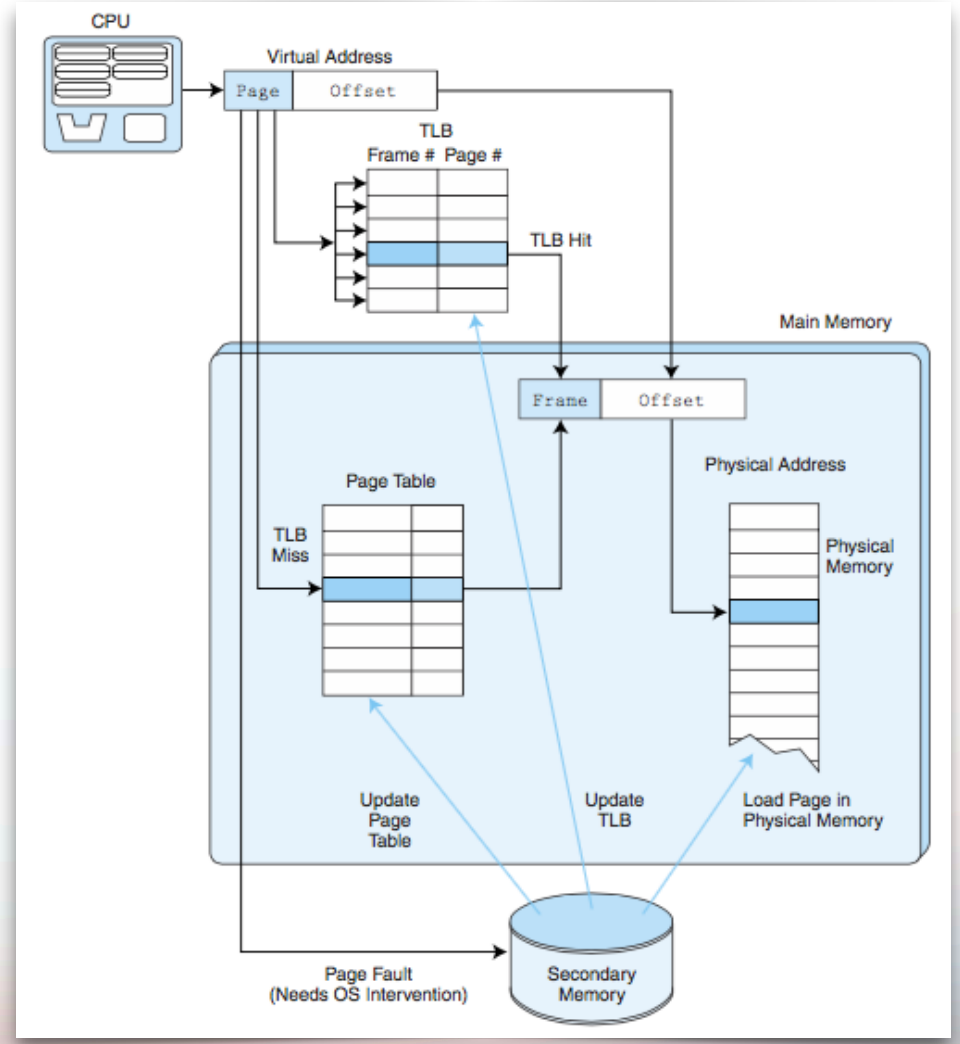
Memoria Virtual - Paginación

¿ Cuánto espacio ocupa la Tabla de página ?
Ejemplo:
32-bit Dirección Virtual, 30-bit Dirección física
12-bit Offset, Cada entrada 4 bytes
2^20 * 4 = 4 MBytes

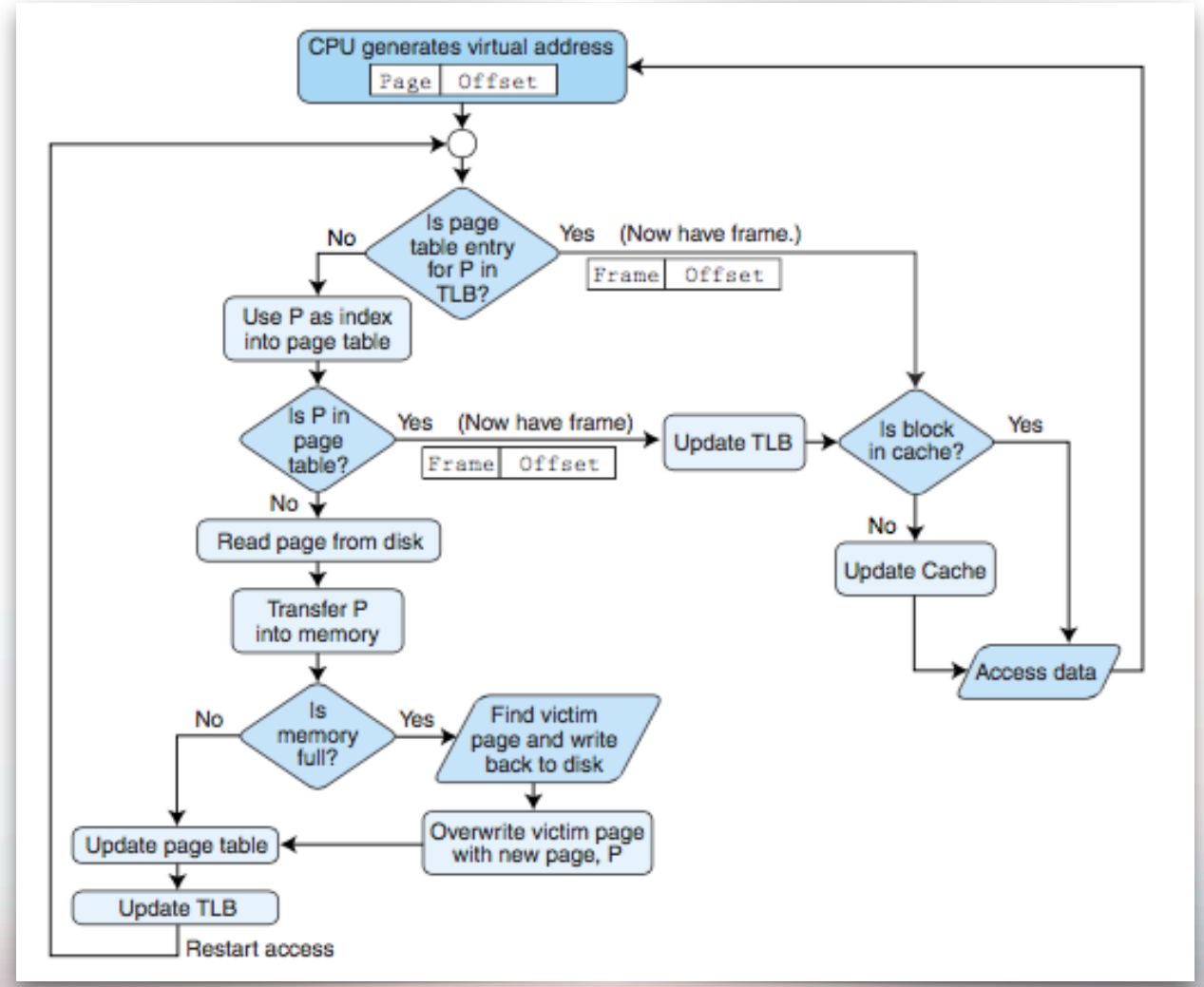
¿ Dónde se almacena ?
En la memoria. Cada acceso a una dirección de memoria (virtual) debe consultar (un acceso memoria) sobre la dirección física.

¿ Cómo lo optimizamos ?
Usando Cache, *Translation Lookaside Buffer (TLB)*

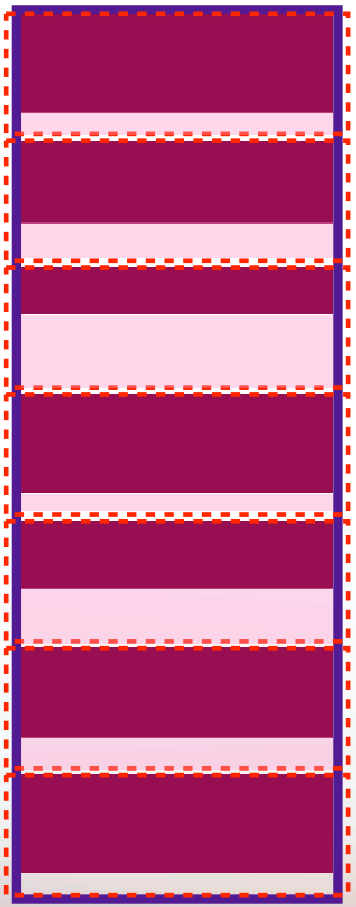
Memoria Virtual - Paginación - Translation Lookaside Buffer (TLB)



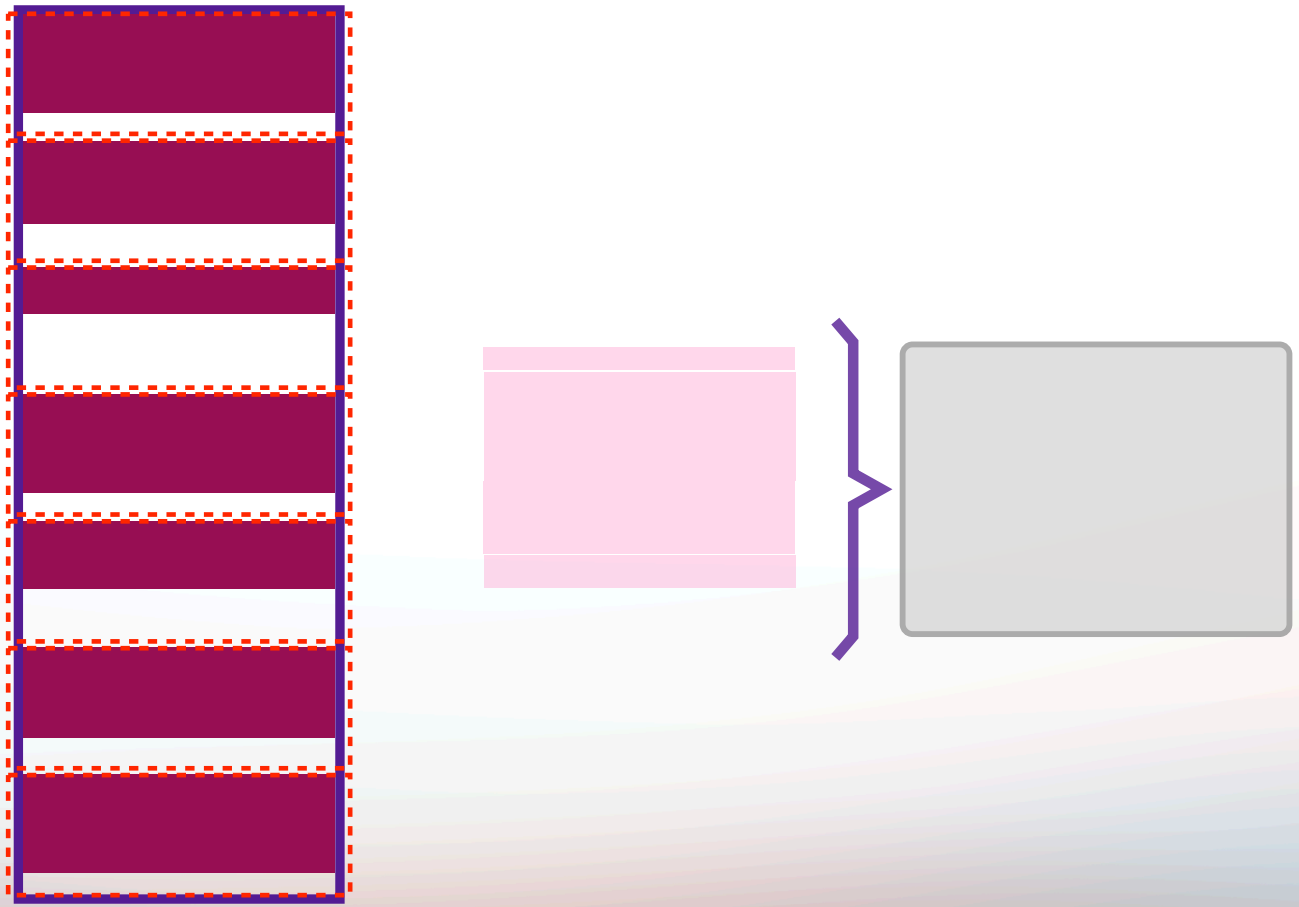
Memoria Virtual - Paginación - Workflow de acceso a memoria



Memoria Virtual - Paginación - Fragmentación Interna



Memoria Virtual - Paginación - Fragmentación Interna



Memoria Virtual (Paginación) vs Memoria Caché

	Memoria Caché	Memoria Virtual
Unidad	Bloque	Página
Falta	Miss	Page Fault
Tamaño	Bloque: 32-64B	Pág.: 4K-16KB
Mapeo	Direct Mapped, N-way Set Associative	Fully Associative
Reemplazo	LRU or Random	LRU
Escritura	Write Through or Back	Write Back
Manejo	Hardware	Hardware(TLB) + Software (S.O.)

Memoria Virtual - Segmentación

- Idea similar a la de Paginación
- Se divide la memoria en **Segmentos** de tamaño **variable.**
- Permite incorporar diferente información como permisos para compartir memoria con otros procesos.
- Se implementa mediante una tabla de segmentos: segmento, inicio, tamaño.
- Sufre de **Fragmentación Externa** (el espacio liberado por otro/s segmentos no es suficiente para incorporar un pedido de segmento más grande)
- En general se utiliza en combinación con Paginado. (Cada segmento se divide en Pág.)