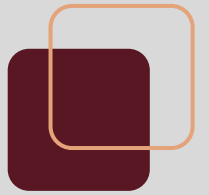
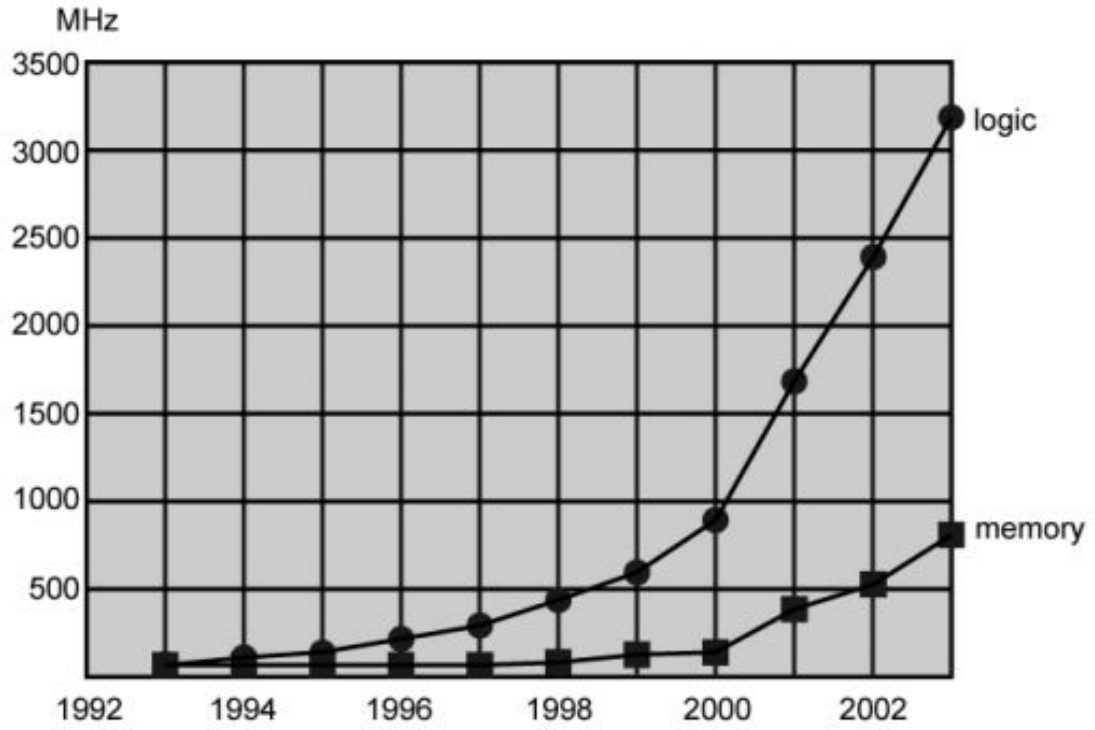


Aspectos avanzados de arquitectura de computadoras
Jerarquía de Memoria

Facultad de Ingeniería - Universidad de la República

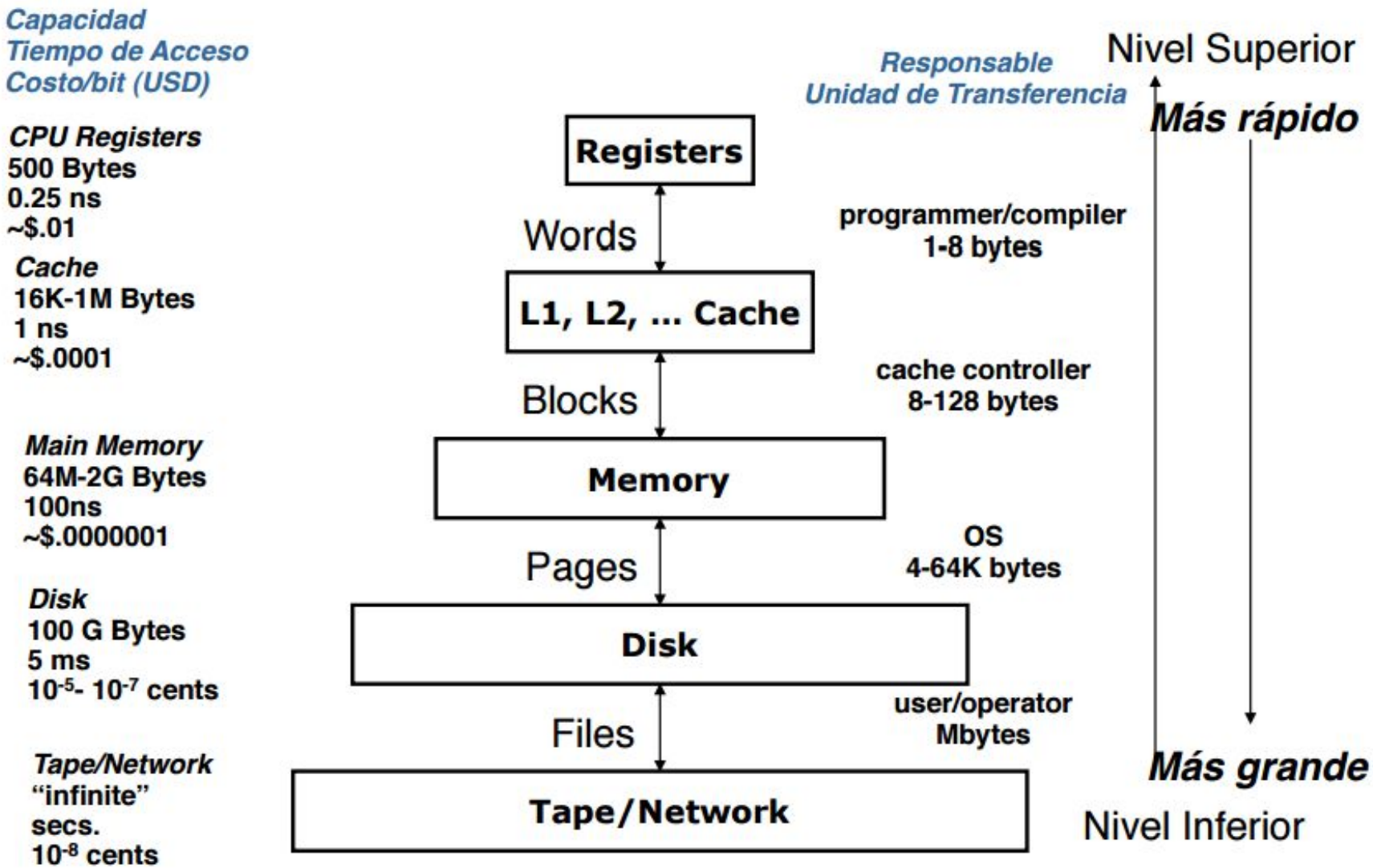
Curso 2017

Tendencias Tecnológicas

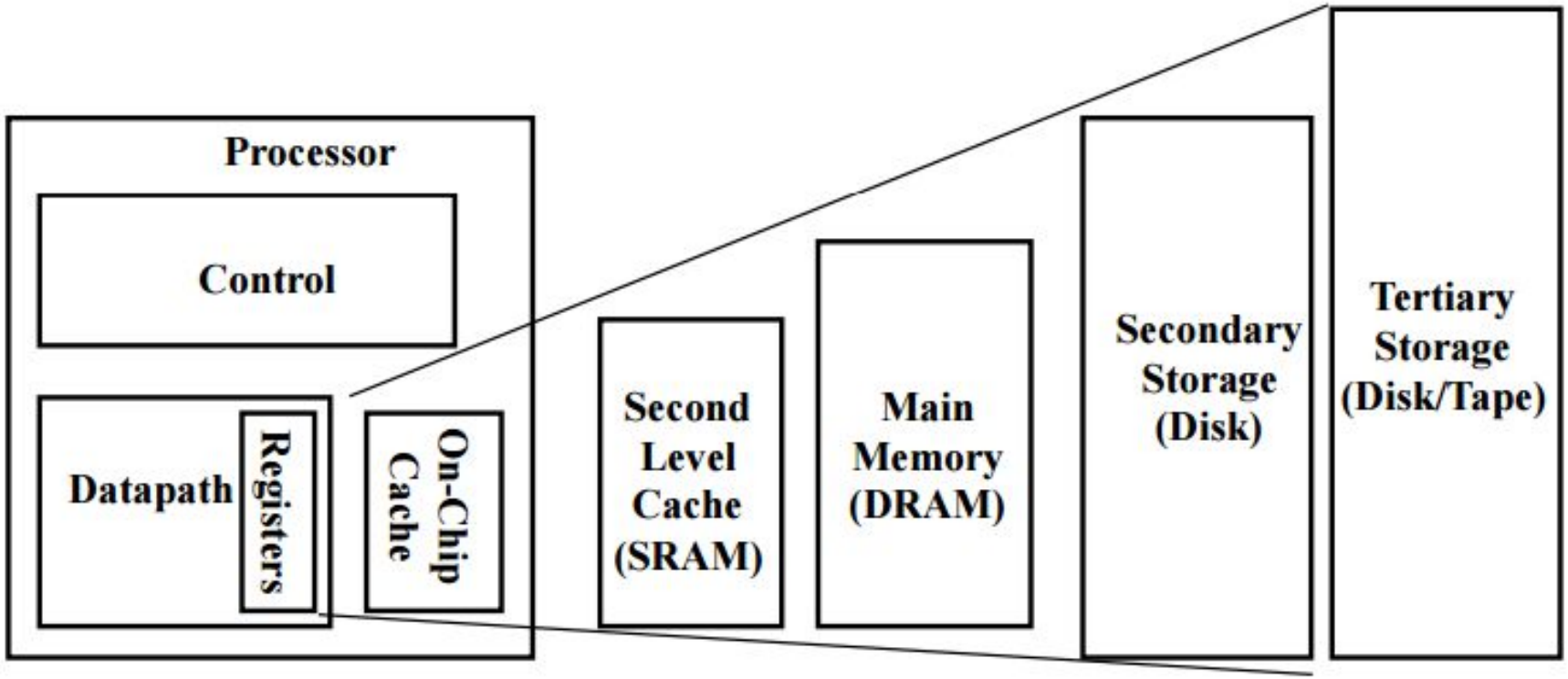


	Capacidad	Velocidad (latencia)
Lógica	2x en 3 años	2x en 1.5 años
DRAM	4x en 3 años	2x en 10 años
Disco	4x en 3 años	2x en 10 años

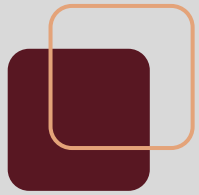
Niveles de Jerarquía de Memoria



Niveles de la Jerarquía de Memoria



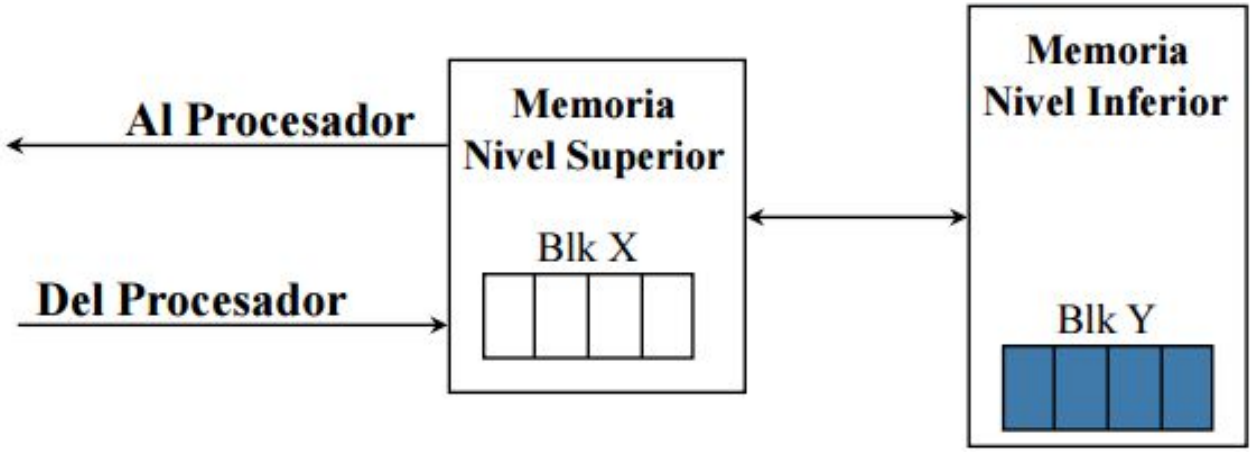
¿Por qué funciona la Jerarquía de Memoria?



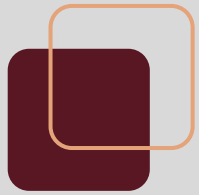
- Principio de Localidad: Los programas acceden a una porción relativamente pequeña del espacio de direcciones en un determinado lapso de tiempo.
  - Localidad Temporal: Si un ítem es referenciado en determinado momento, es común que vuelva a ser referenciado poco tiempo después.
  - Localidad Espacial: Cuando un ítem es referenciado en determinado momento, es común que algunos ítems con direcciones "cercanas" también sean accedidos poco tiempo después.

Jerarquía de Memoria
¿Cómo funciona?

- Localidad Temporal: Mantener los datos más recientemente accedidos "cercanos" al procesador.
- Localidad Espacial: Mover bloques de palabras contiguas al nivel superior.

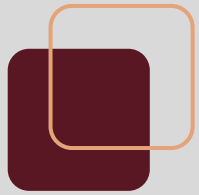


¿Cómo se maneja la jerarquía?



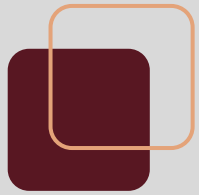
- Registros <-> Memoria
  - Por el compilador (programador?)
- Caché <-> Memoria
  - Por el hardware
- Memoria <-> Discos
  - Por el hardware y el sistema operativo (memoria virtual)
  - Por el programador (archivos)

Jerarquía de Memoria: Terminología (1/2)



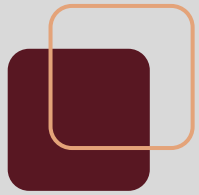
- Hit: los datos están en algún bloque del nivel superior
  - Hit Rate: fracción de accesos a memoria encontrados en el nivel superior
  - Hit Time: Tiempo de acceso al nivel superior
    - Tiempo para determinar hit/miss + tiempo de lectura/escritura

Jerarquía de Memoria: Terminología (2/2)



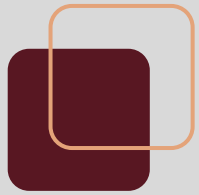
- Miss: los datos deben ser traídos desde un bloque en el nivel inferior
  - Miss Rate = 1 - (Hit Rate)
  - Miss Penalty: Tiempo adicional requerido en caso de Miss.
- Hit Time << Miss Penalty
- Tiempo medio de acceso a memoria
  - Hit time + Miss Rate * Miss Penalty

Tecnologías de Memoria (1/2)



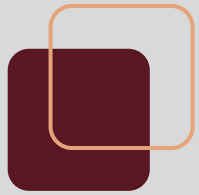
- Acceso “Aleatorio”
  - El tiempo de acceso es el mismo para cualquier posición
  - DRAM: Dynamic Random Access Memory
    - Alta densidad, baja potencia, barata, lenta
    - Dinámica: necesita ser “refrescada” regularmente
  - SRAM: Static Random Access Memory
    - Baja densidad, alta potencia, cara, rápida
    - Estática: su contenido no se pierde mientras se mantenga la alimentación

Tecnologías de Memoria (2/2)



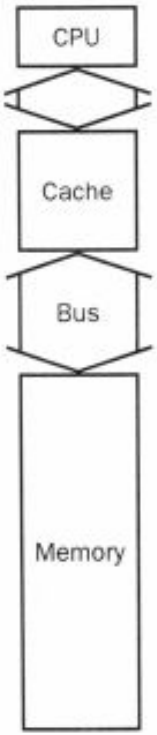
- Tecnología de Acceso “Non-so-random”
  - Tiempo de acceso varía según la posición y el momento
  - Ejemplos: Disco, CDROM
- Tecnología de Acceso Secuencial:
  - Tiempo de acceso lineal con la posición (ej. Cinta)

Algunas medidas de rendimiento de memoria

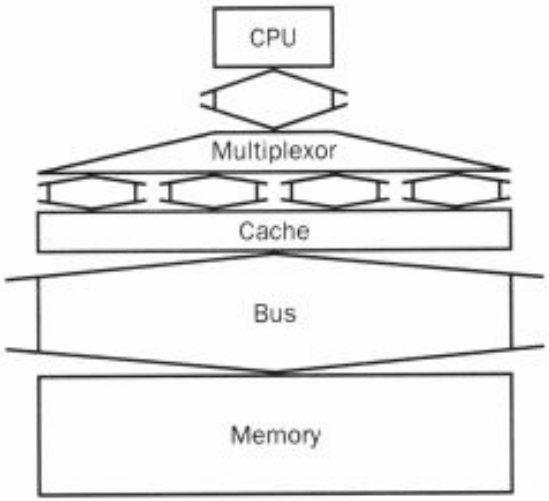


- Tiempo de Acceso
  - Tiempo transcurrido entre presencia de direcciones y obtención de datos válidos
- Tiempo de Ciclo de Memoria
  - Tiempo entre dos accesos consecutivos
  - Tiempo de Ciclo: acceso + recuperación
- Tasa de Transferencia
  - Tasa (Bits/Segundo) de transferencia de datos

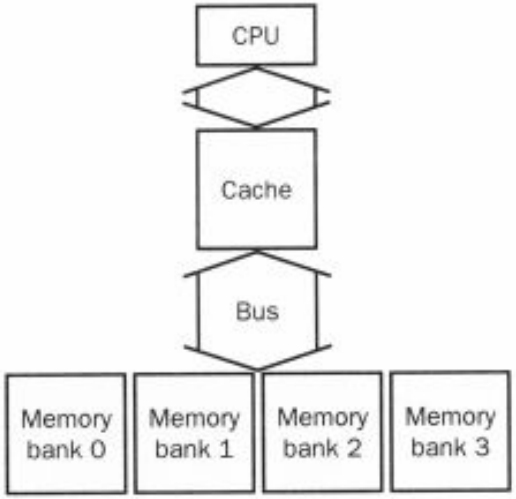
Memoria Principal: La organización puede mejorar el rendimiento



a. One-word-wide memory organization



b. Wide memory organization

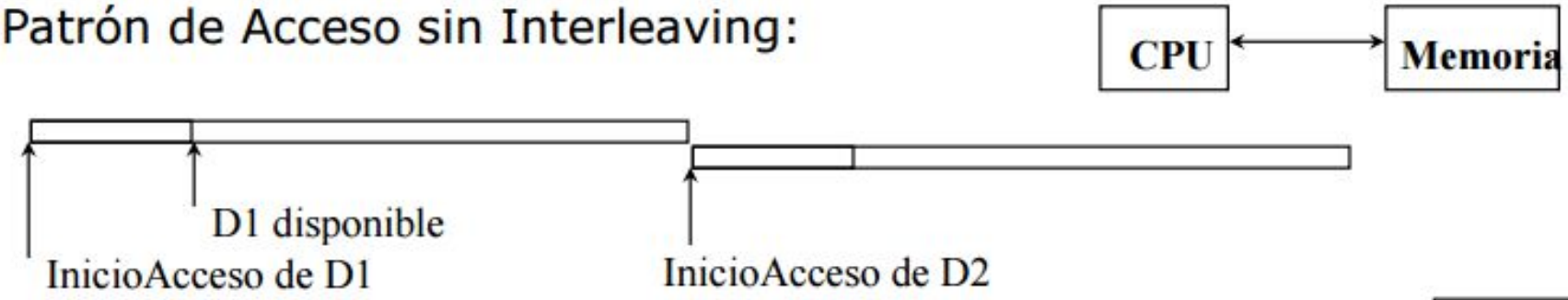


c. Interleaved memory organization

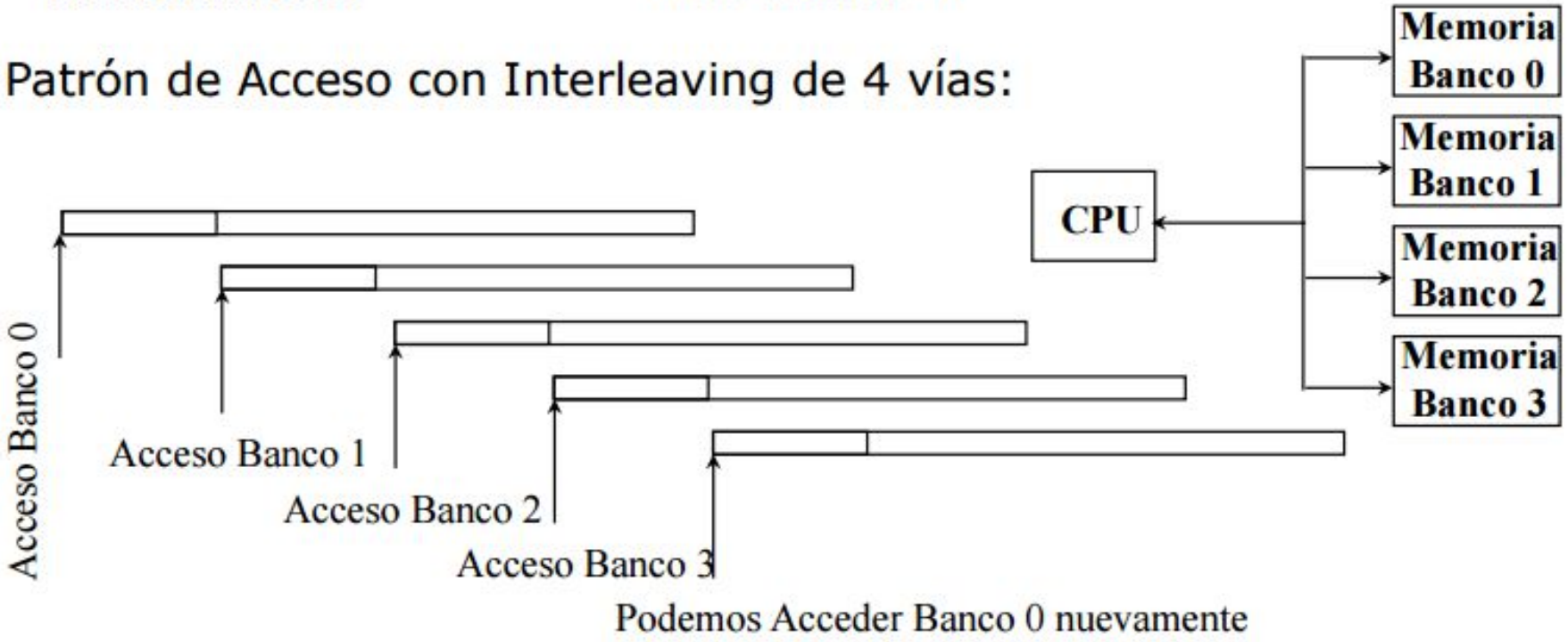
- Simple:
  - CPU, Cache, Bus, Memoria mismo ancho(32 bits)
- Ancho (Wide):
  - CPU/Mux 1 palabra; Mux/Cache, Bus, Memoria N palabras
- Entrelazado (Interleaved):
  - CPU, Cache, Bus 1 palabra: Memoria N Módulos; el ejemplo es de 4 módulos, word interleaved (estrelazado de palabra)

Entrelazado (Interleaving)

Patrón de Acceso sin Interleaving:

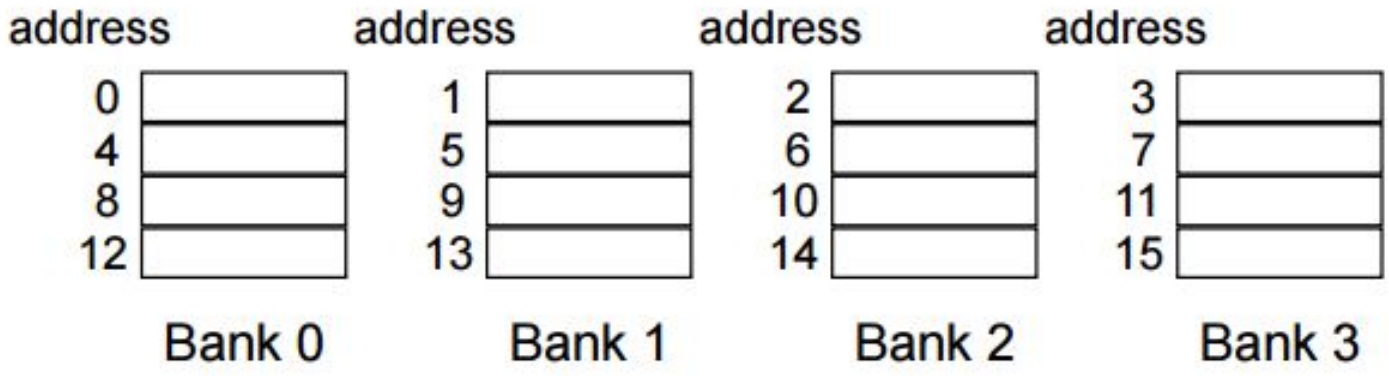


Patrón de Acceso con Interleaving de 4 vías:



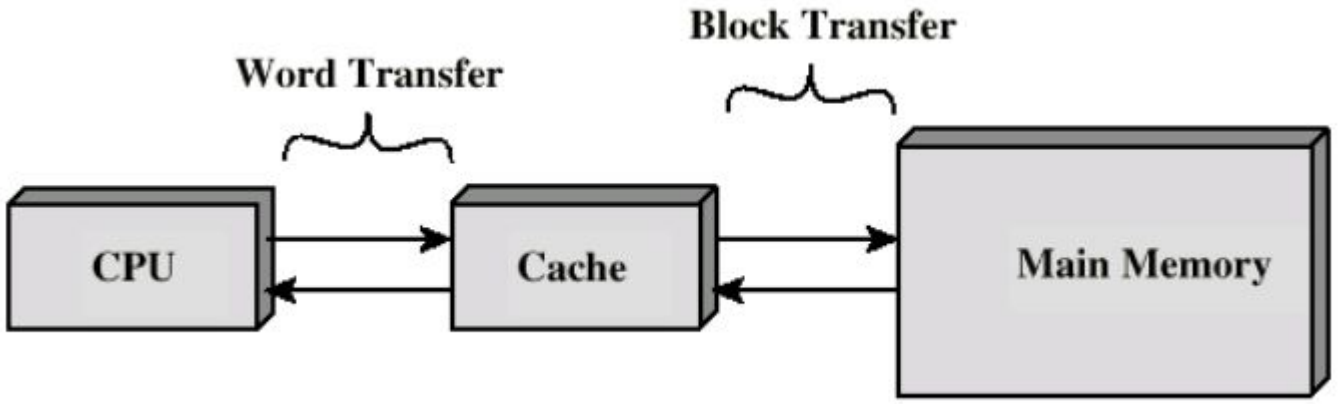
Rendimiento de la Memoria Principal

- Modelo de Tiempos
  - 1 u. para enviar direcciones
  - 10 u. de tiempo de acceso
  - 1 u. para enviar datos
  - Cache con Bloque de 4 palabras
    - Simple = 4 x (1+10+1) = 48
    - Ancho = 1 + 10 + 1 = 12
    - Entrelazado = 1 + 10 + 1+ 1 + 1 + 1 = 15

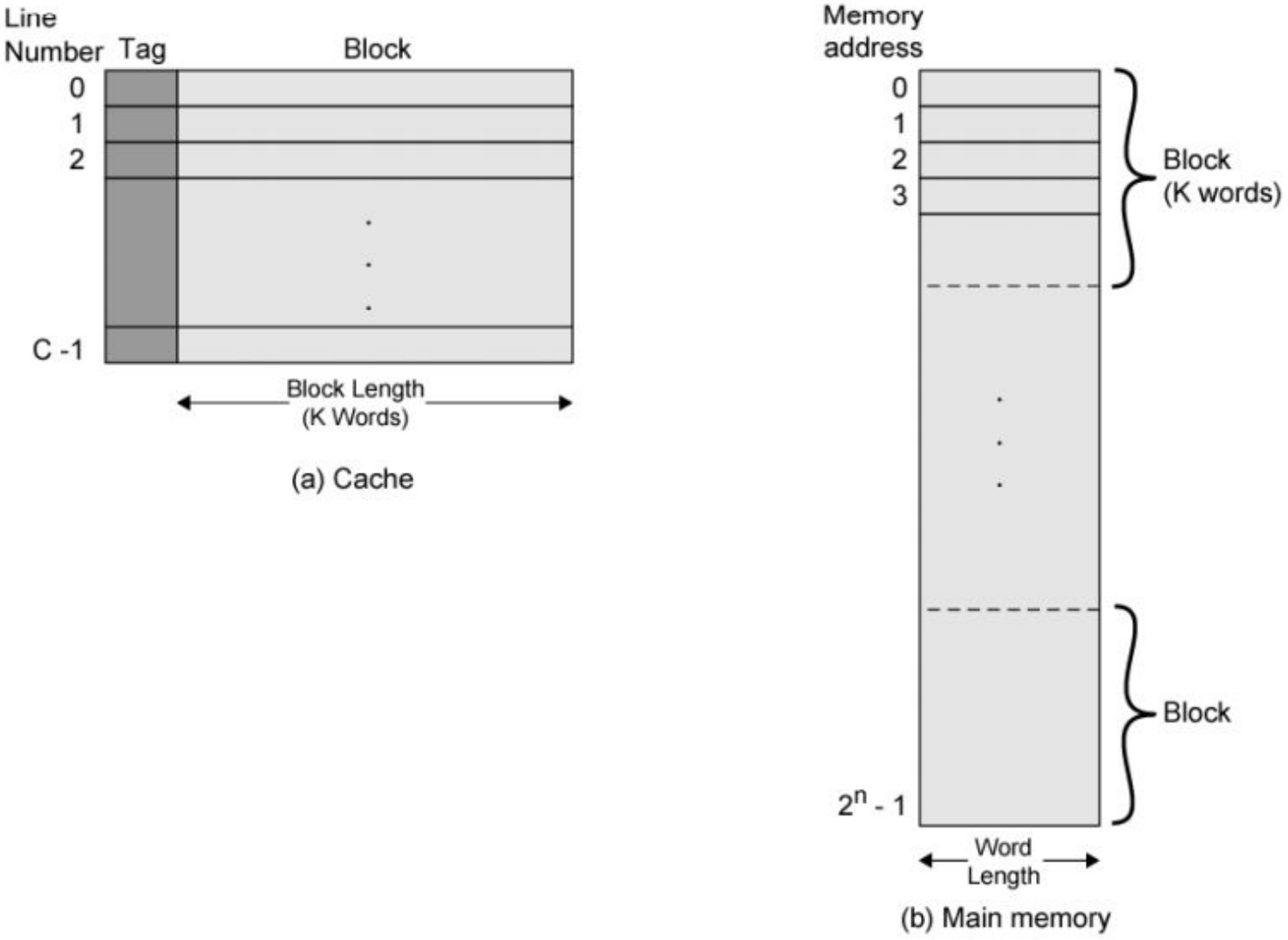


Memoria Caché

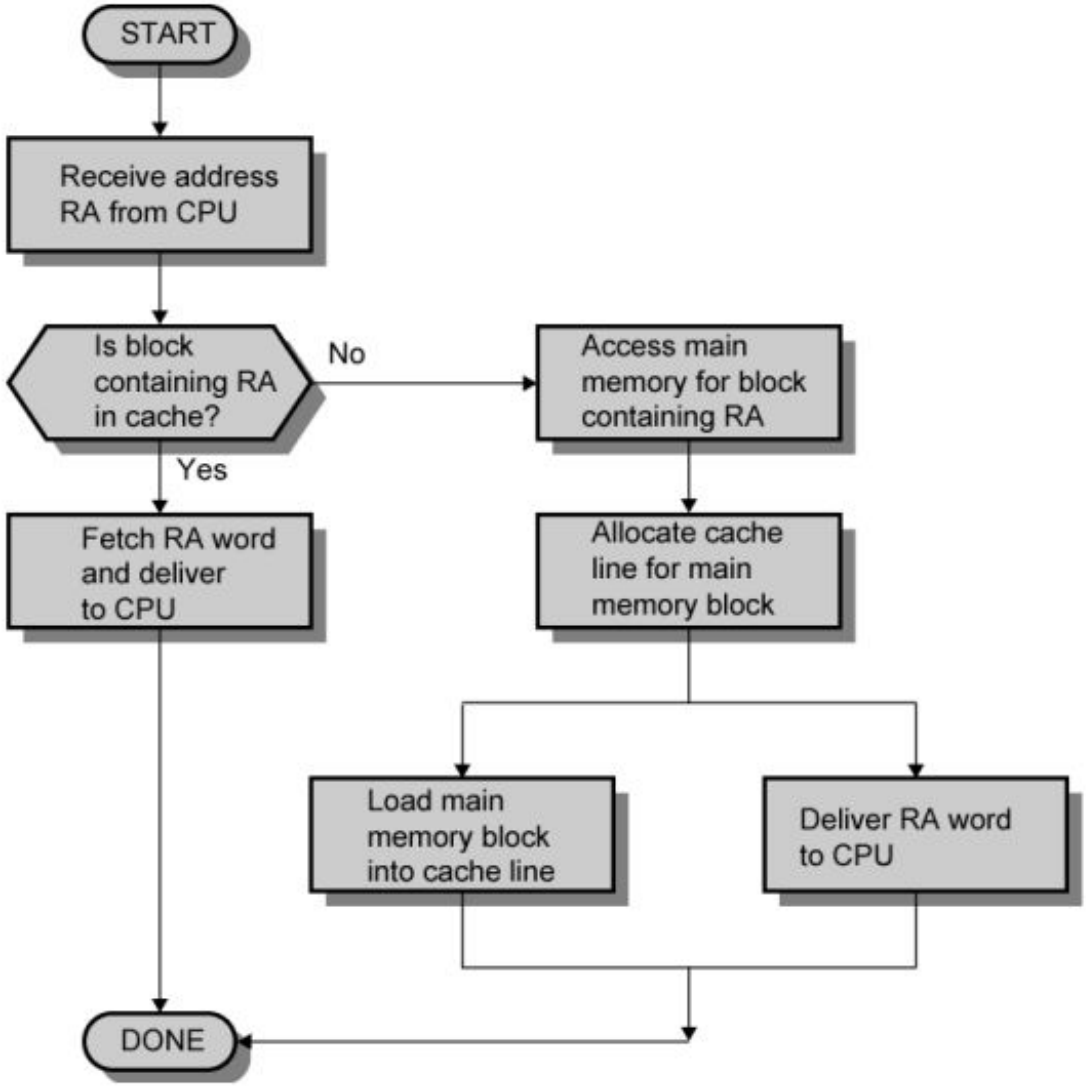
- Memoria rápida, escasa
- Se instala entre la memoria principal y la CPU
- Puede esta en el chip de la CPU (L1) o módulo externo (L2)
- La memoria principal se accede por bloques de K palabras (K > 1). Cuando se lee un bloque, se almacena en una línea de caché.



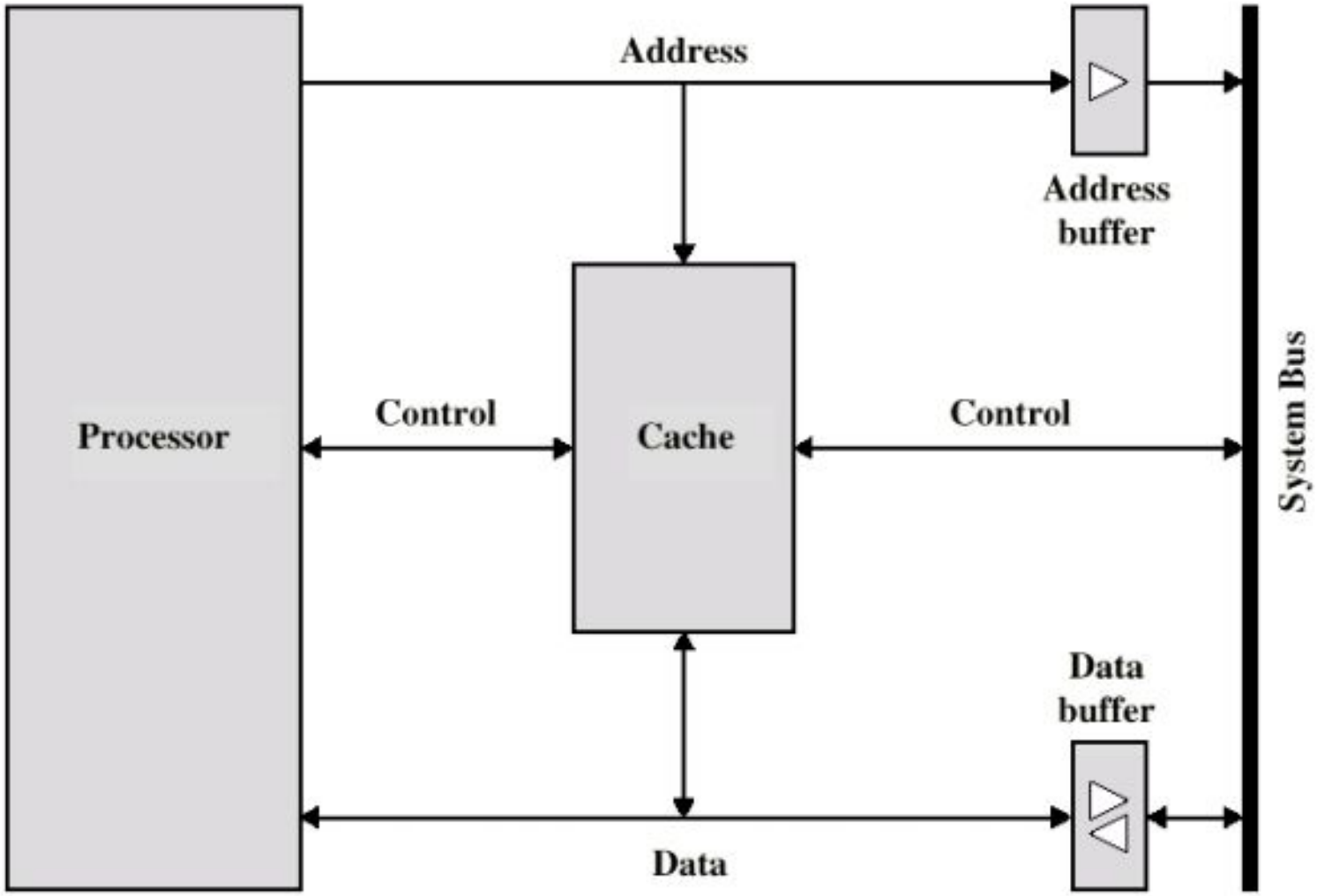
Estructura de la Caché/Memoria



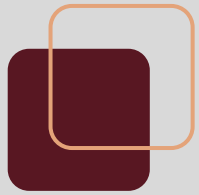
Operativa de la cache: Lectura



Organización típica del caché

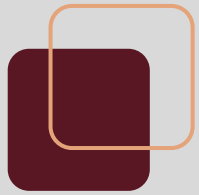


Diseño de la Caché (1/2)



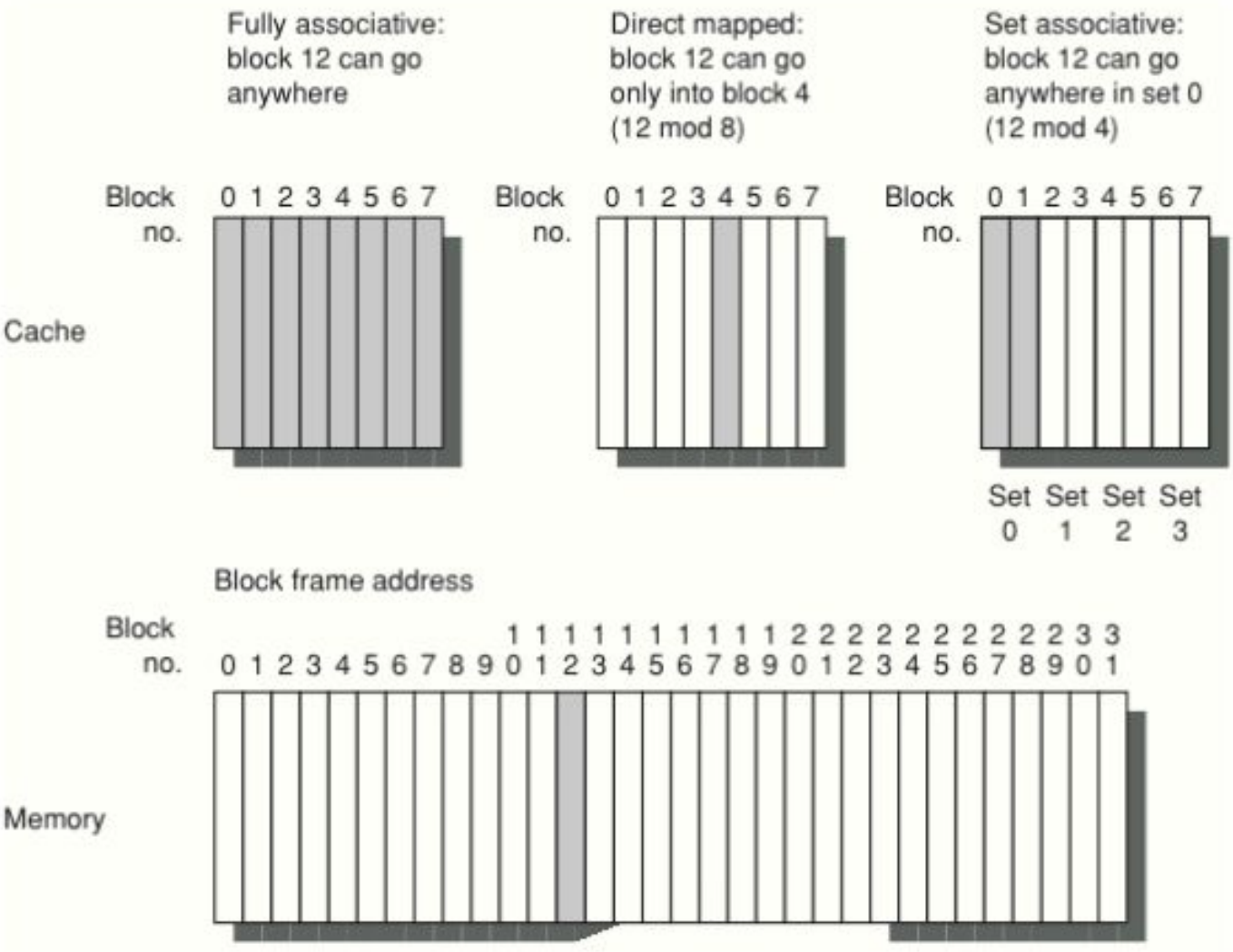
- Tamaño: Compromiso Costo/Velocidad
  - Más caché, más velocidad (hasta cierto punto)
- Función de correspondencia (Mapping Function): Entre bloques de memoria principal y líneas del caché
- Algoritmo de sustitución: ¿Qué bloque se desecha cuando se lee un bloque nuevo?
- Política de escritura

Diseño de la Caché (2/2)

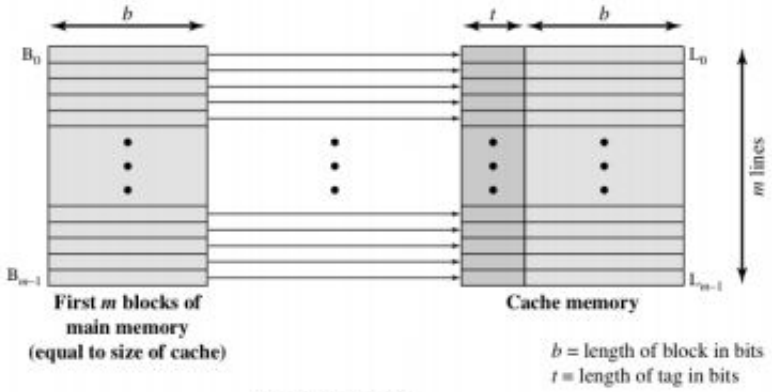


- Tamaño del bloque:
  - Para un tamaño de caché dado: ¿más líneas o líneas más grandes?
- Cantidad de cachés:
  - ¿Caché unificada o cachés separadas para datos e instrucciones?

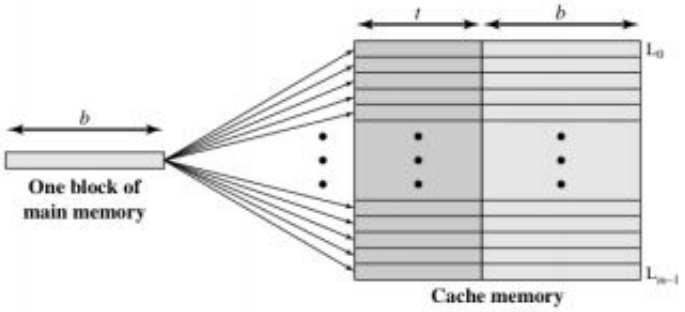
Función de correspondencia (1/6)



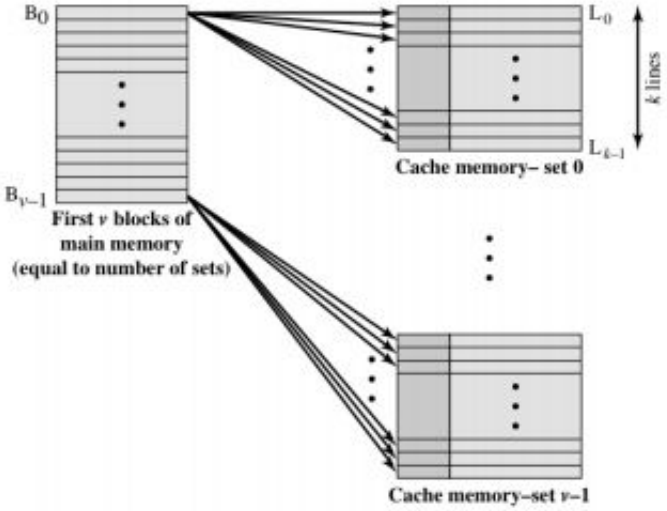
Función de correspondencia (2/6)



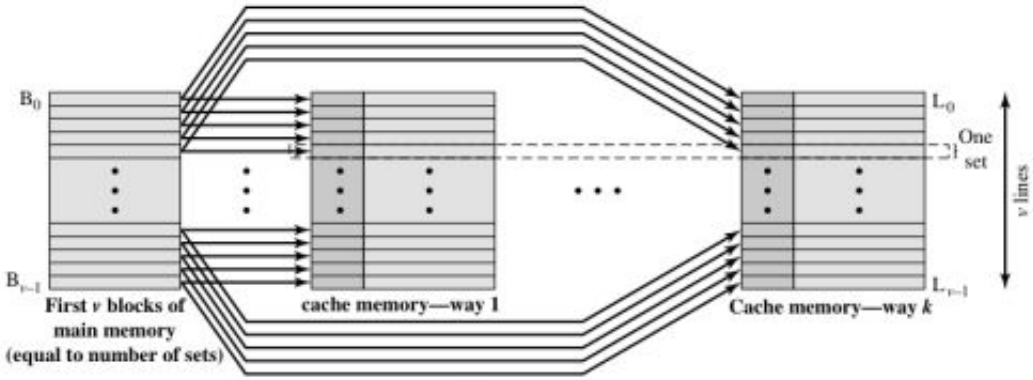
(a) Direct mapping



(b) Associative mapping

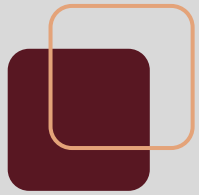


(a) v Associative–mapped caches



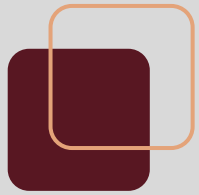
(b) k Direct–mapped caches

Función de correspondencia (3/6)



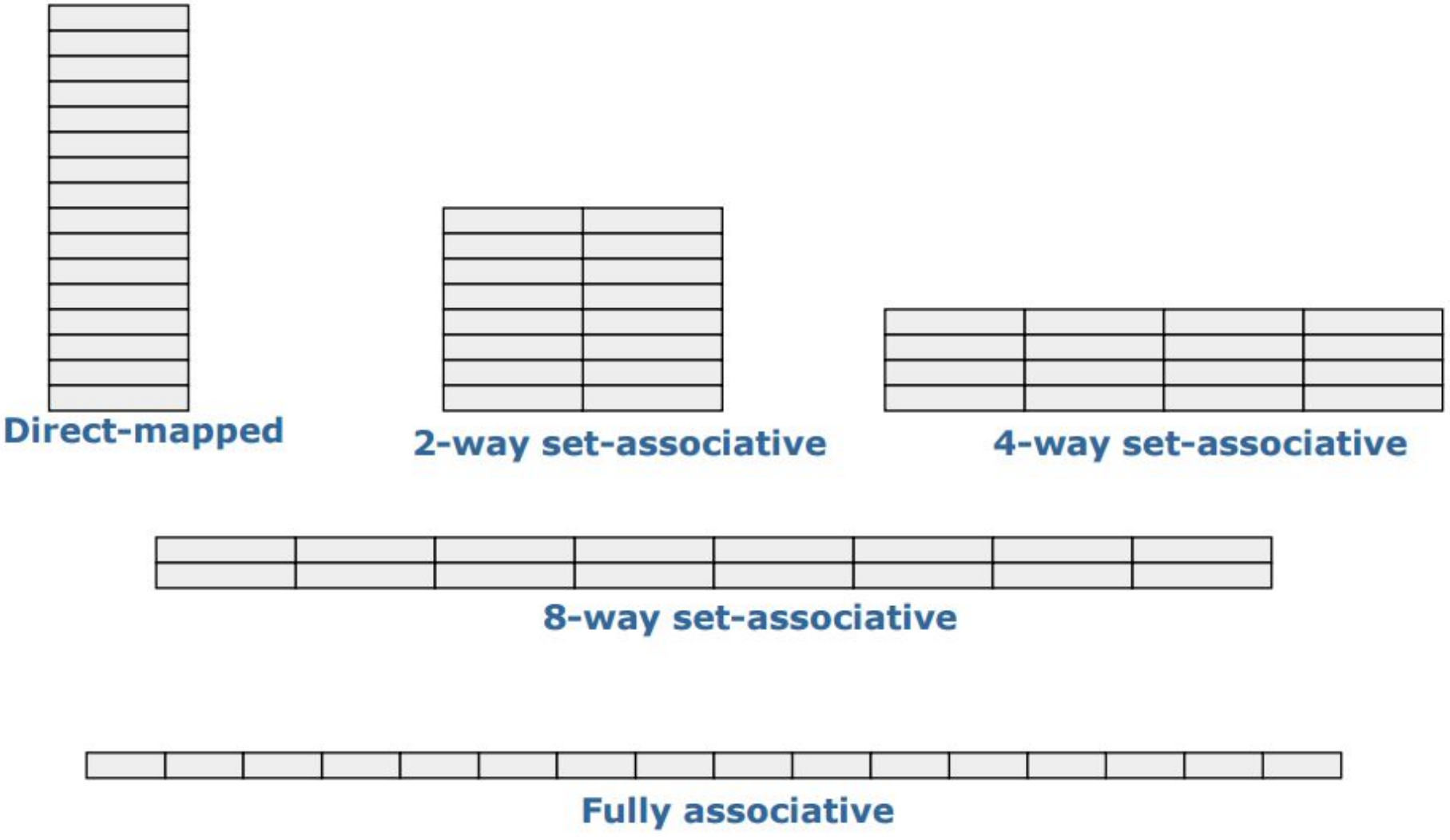
- Correspondencia Directa (Direct Mapped Cache)
  - Cada bloque de memoria solo puede alojarse en una línea del caché
  - No hay que tomar decisiones de reemplazo: Bloque actual reemplaza bloque anterior en cada línea del caché
- Correspondencia Totalmente Asociativa (Fully Associative Cache)
  - Cada bloque de memoria se puede alojar en cualquier línea del caché

Función de correspondencia (4/6)

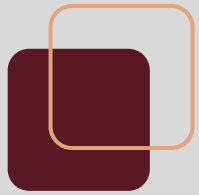


- Correspondencia Asociativa por Conjuntos (N-way Set Associative Cache)
  - Cada bloque de memoria tiene N opciones de localización en el caché
- Caché miss en el "N-way Set Associative" o "Fully Associative" Cache
  - Se debe traer un nuevo bloque desde memoria
  - Se debe descartar un bloque del caché para hacer lugar
  - Hay que decidir qué bloque descartar!

Función de correspondencia (5/6)

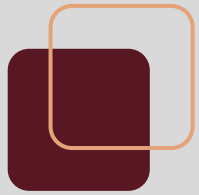


Función de correspondencia (6/6)



- Para todos los casos consideraremos el mismo ejemplo:
  - Cache de 64 KBytes
  - Bloques de 4 bytes
  - Caché de 16K (214) líneas de 4 bytes cada una
  - 16 MBytes de memoria principal
  - 24 bits de direcciones (224=16 MB)

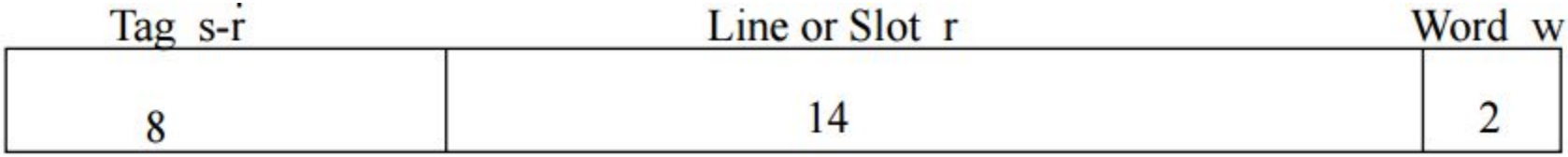
Correspondencia Directa (1/2)



- Cada bloque de memoria principal se corresponde a una línea de la caché
- Se considera la dirección dividida en dos partes
  - w bits menos significativos identifican una palabra única dentro del bloque
  - s bits más significativos especifican un bloque de memoria
- Los s bits de bloque se dividen en un campo línea de caché, de r bits, y un tag de s-r bits (más significativos)

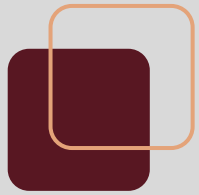
Correspondencia Directa (2/2)

- En nuestro ejemplo, dirección de 24 bits:
  - 2 bits de identificador de palabra (bloques de 4 bytes)
  - 22 bits de identificador de bloque
    - Campo Tag de 8 bits (=22-14)
    - Campo Line o Slot de 14 bits



- Bloques en la misma línea tienen tag único. Se chequea el contenido de la caché mediante el tag de línea

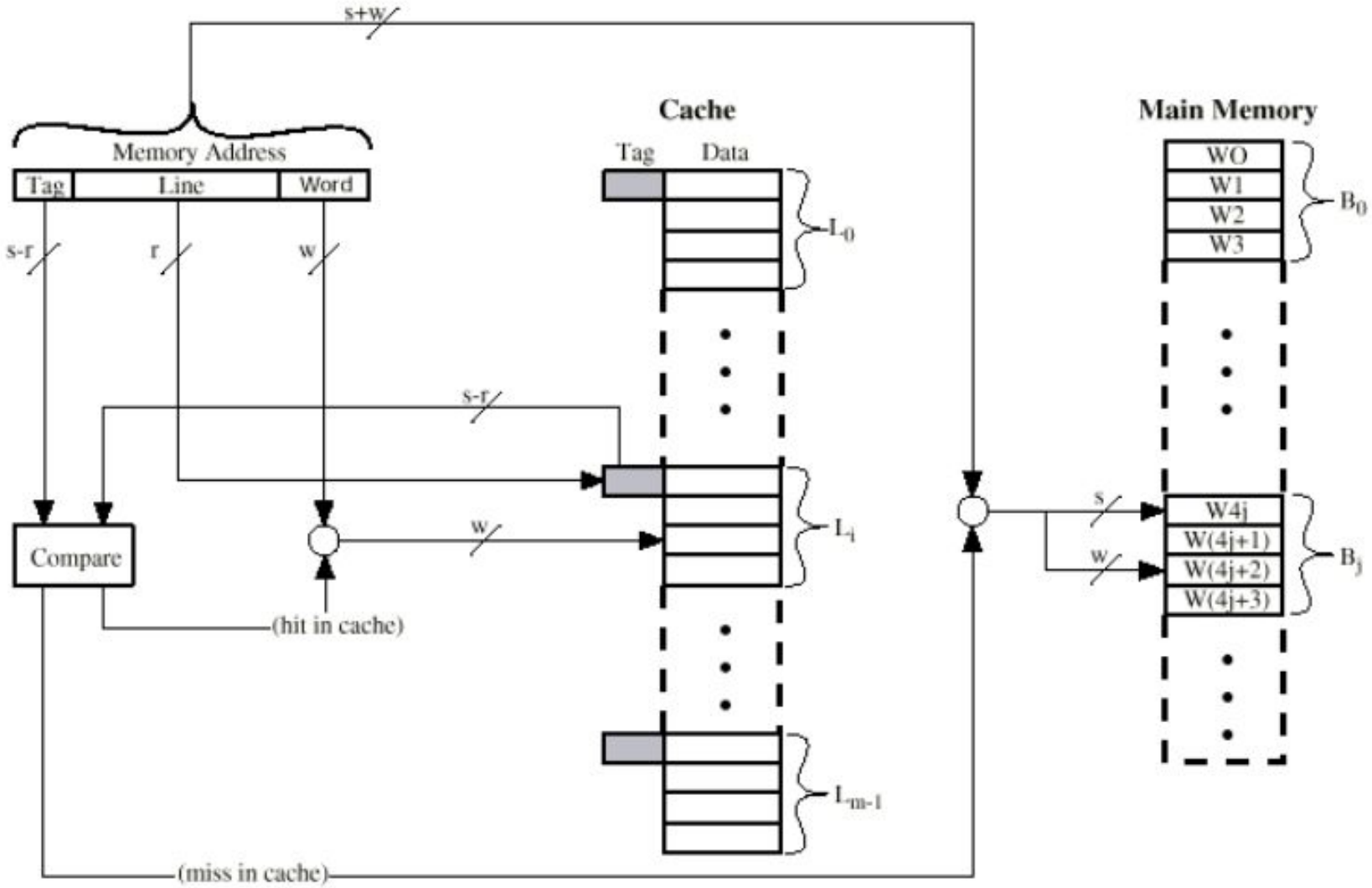
Tabla de líneas de la caché con Correspondencia Directa



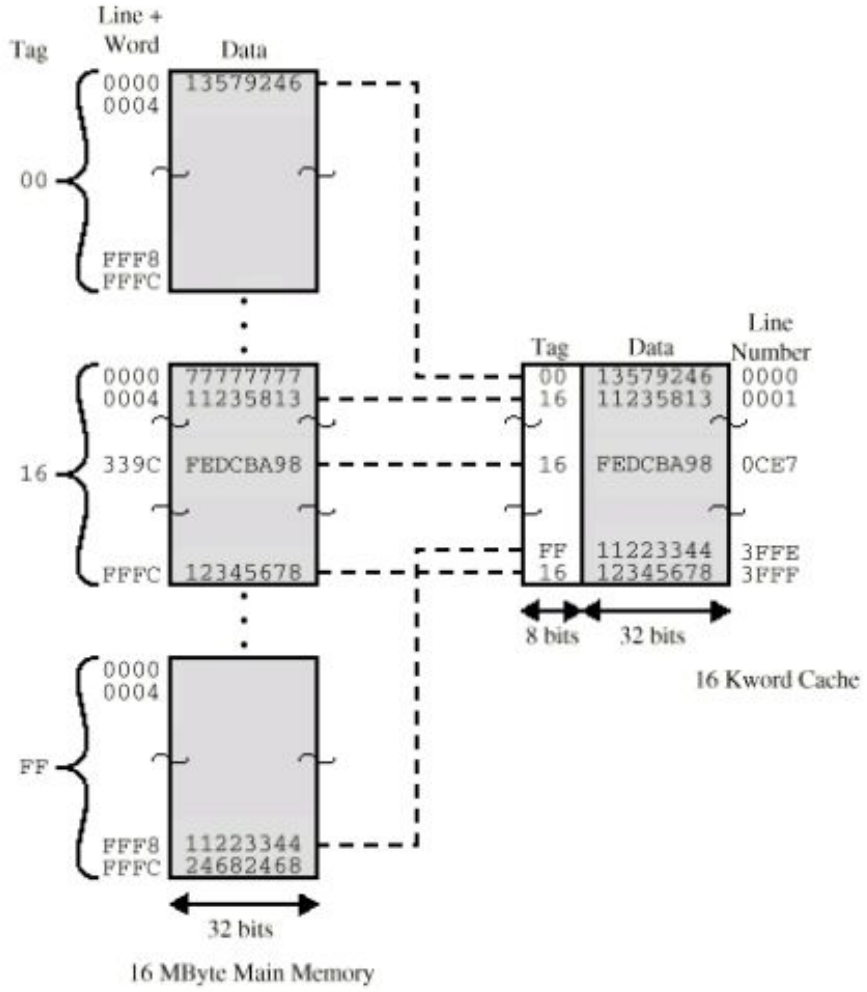
- Correspondencia:
  - i=j módulo m
  - i línea del caché
  - j número de bloque de memoria principal
  - m=2^r (cantidad de líneas del caché)

Línea de Caché	Bloque contenido
0	0, m, 2m, 3m…2s-m
1	1,m+1, 2m+1…2s-m+1
…	…
m-1	m-1, 2m-1,3m-1…2s-1

Organización de la caché con correspondencia directa

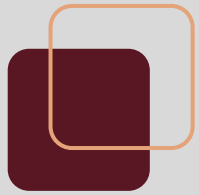


Ejemplo correspondencia directa



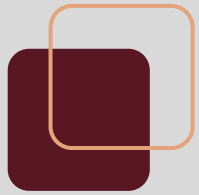
Tag : 8 bits
Line : 14 bits
Word : 2 bits

Correspondencia Totalmente Asociativa

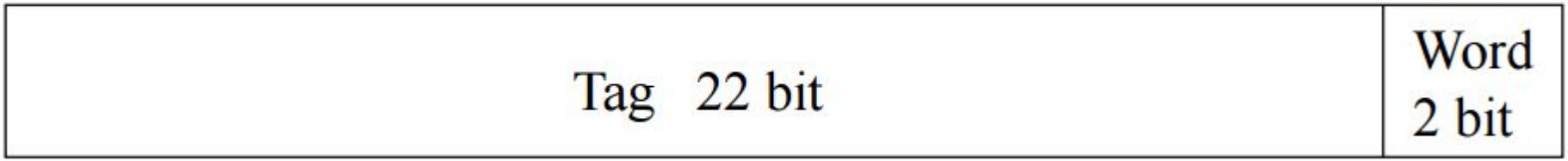


- Un bloque de memoria principal se puede cargar en cualquier línea de la cach
- Dirección de memoria se interpreta como un tag y un word
- Cada tag identifica unívocamente un bloque de memoria
- Cada tag debe ser examinado para ver si hay coincidencia
  - Búsqueda se encarece (tiempo)

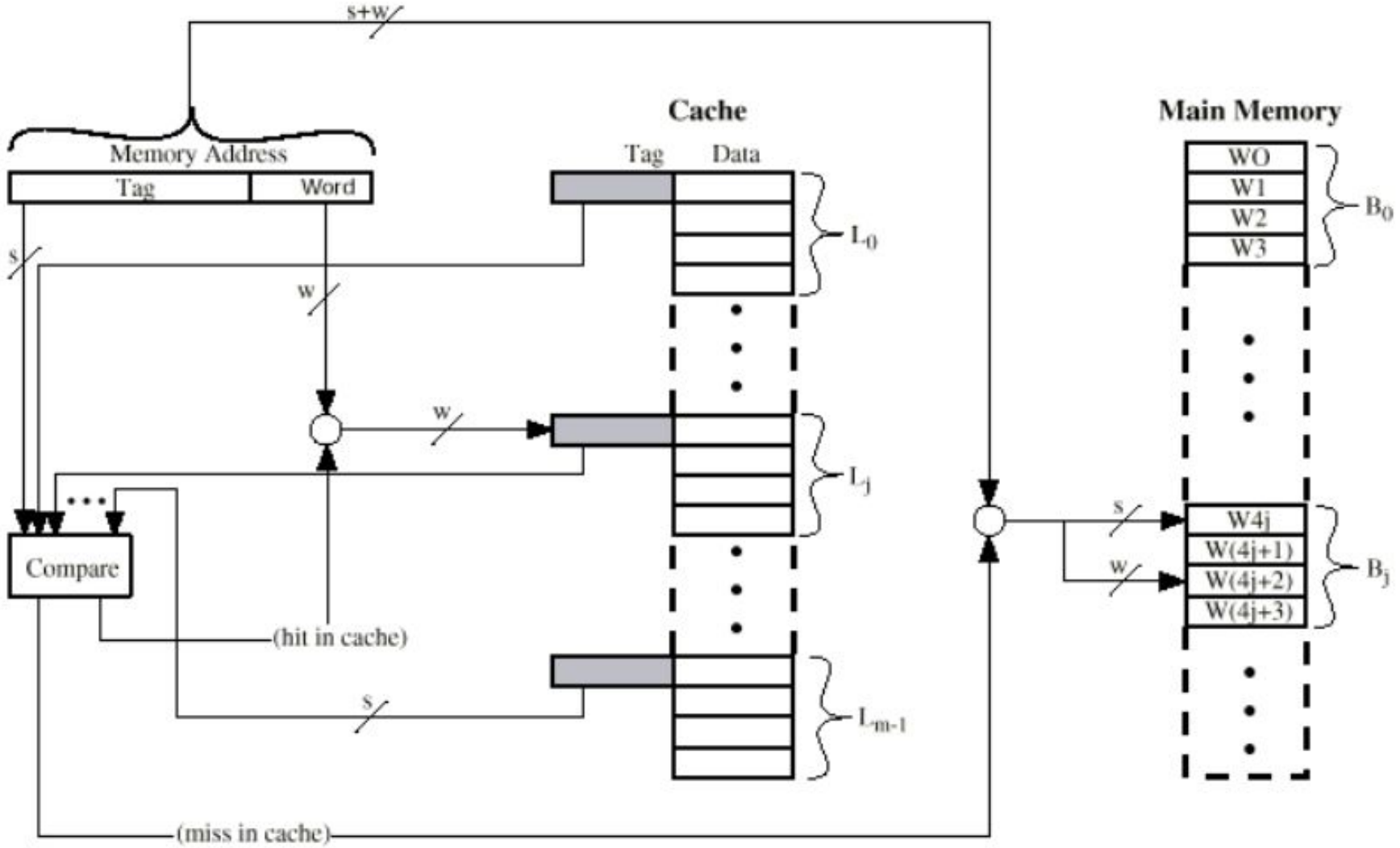
Estructura de direccionamiento correspondencia asociativa



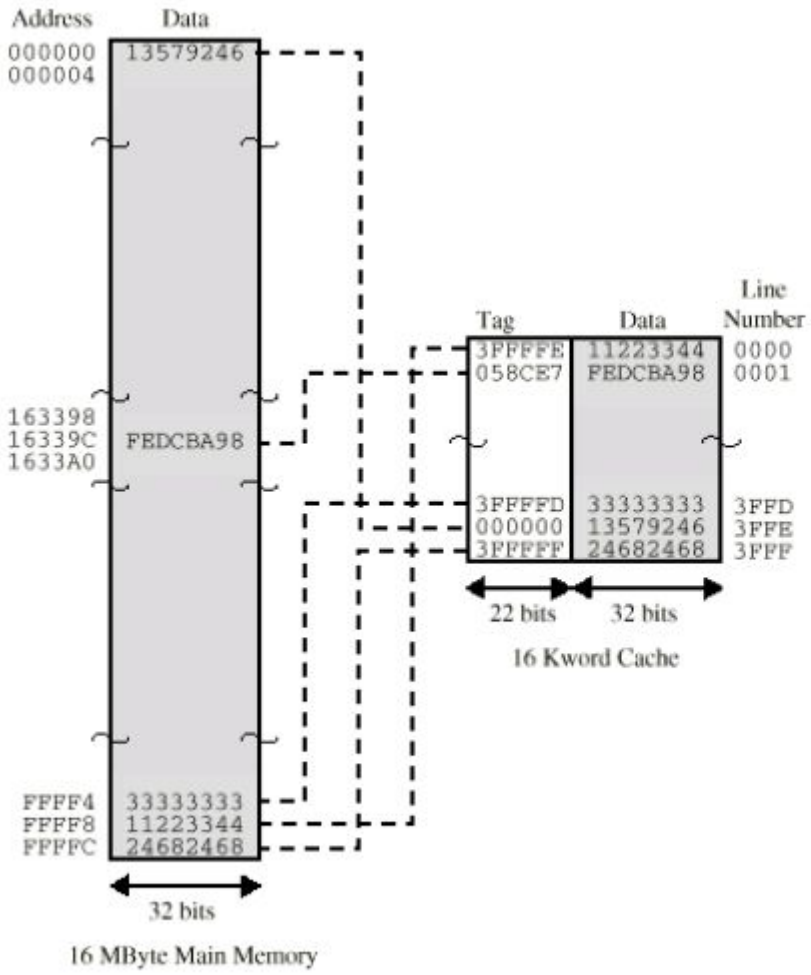
- En nuestro ejemplo:
  - Tag de 22 bits almacenada con cada bloque de datos de 32 bits
  - Se compara el campo tag con la tag guardada en la cache para chequear un hit
  - 2 bits menos significativos de la dirección identifican la palabra



Organización de la caché con correspondencia fully associative

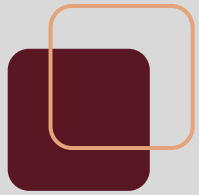


Ejemplo correspondencia totalmente asociativa



Tag : 22 bits
Word : 2 bits

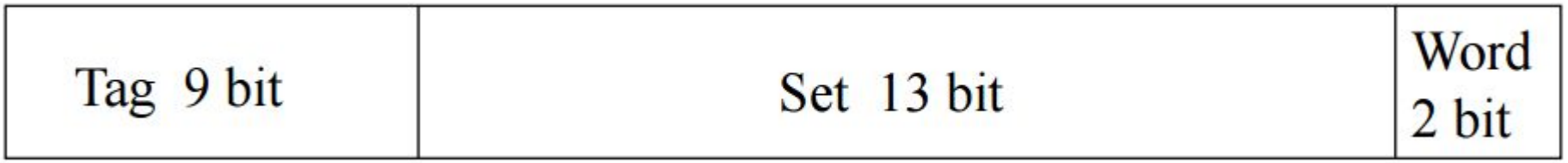
Correspondencia asociativa por conjuntos (Set Associative)



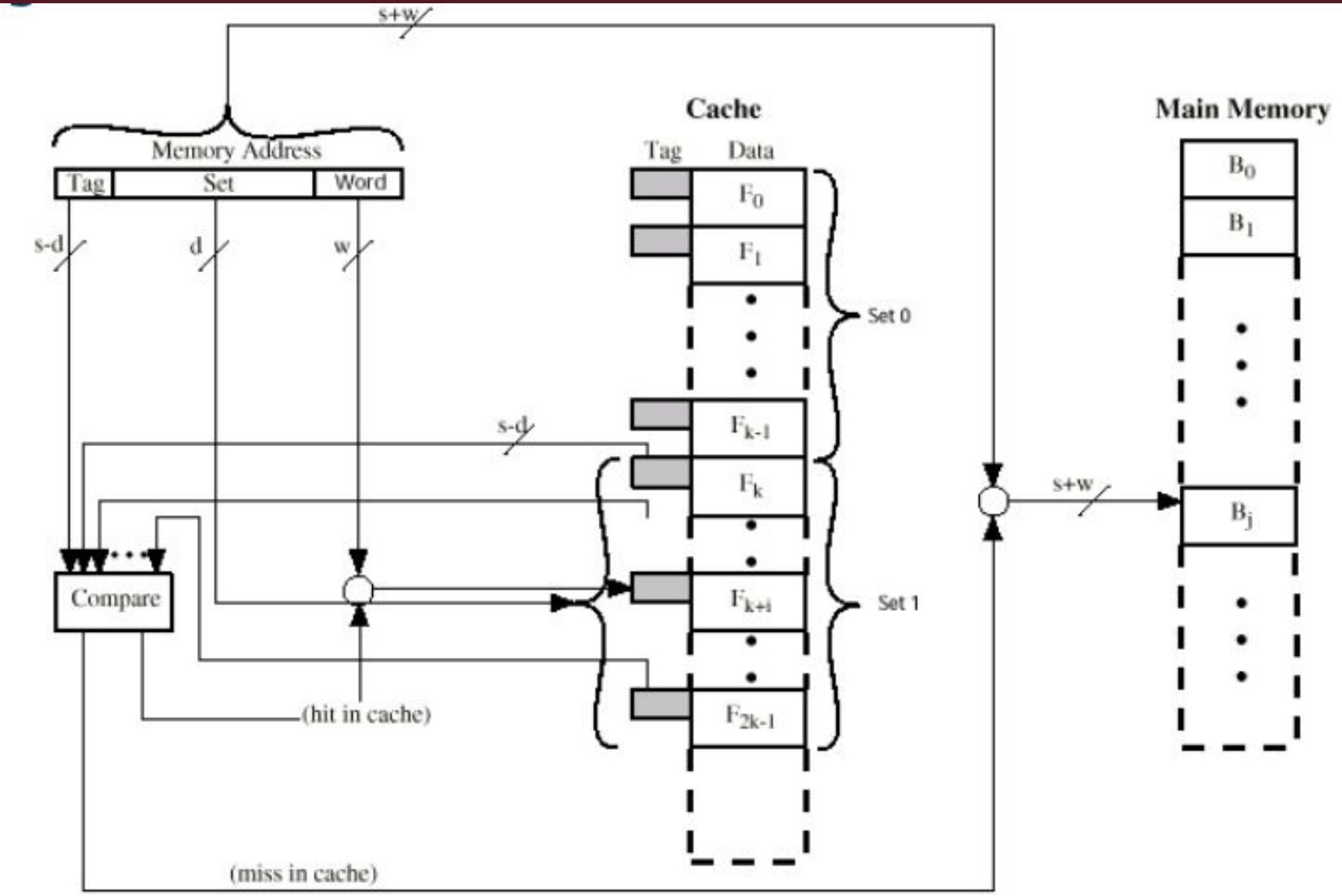
- Caché dividida en conjuntos
  - Cada conjunto contiene una cantidad fija de líneas
  - Un bloque se asocia a un conjunto fijo, pero puede almacenarse en cualquiera de sus líneas
  - Bloque B puede estar en cualquier línea del conjunto i
  - Ejemplo 2 líneas por conjunto
    - Correspondencia asociativa por conjuntos de dos vías.
    - Un bloque determinado puede estar en cualquiera de las dos líneas del conjunto que le corresponde

Estructura de direccionamiento "Set Associative Mapping"

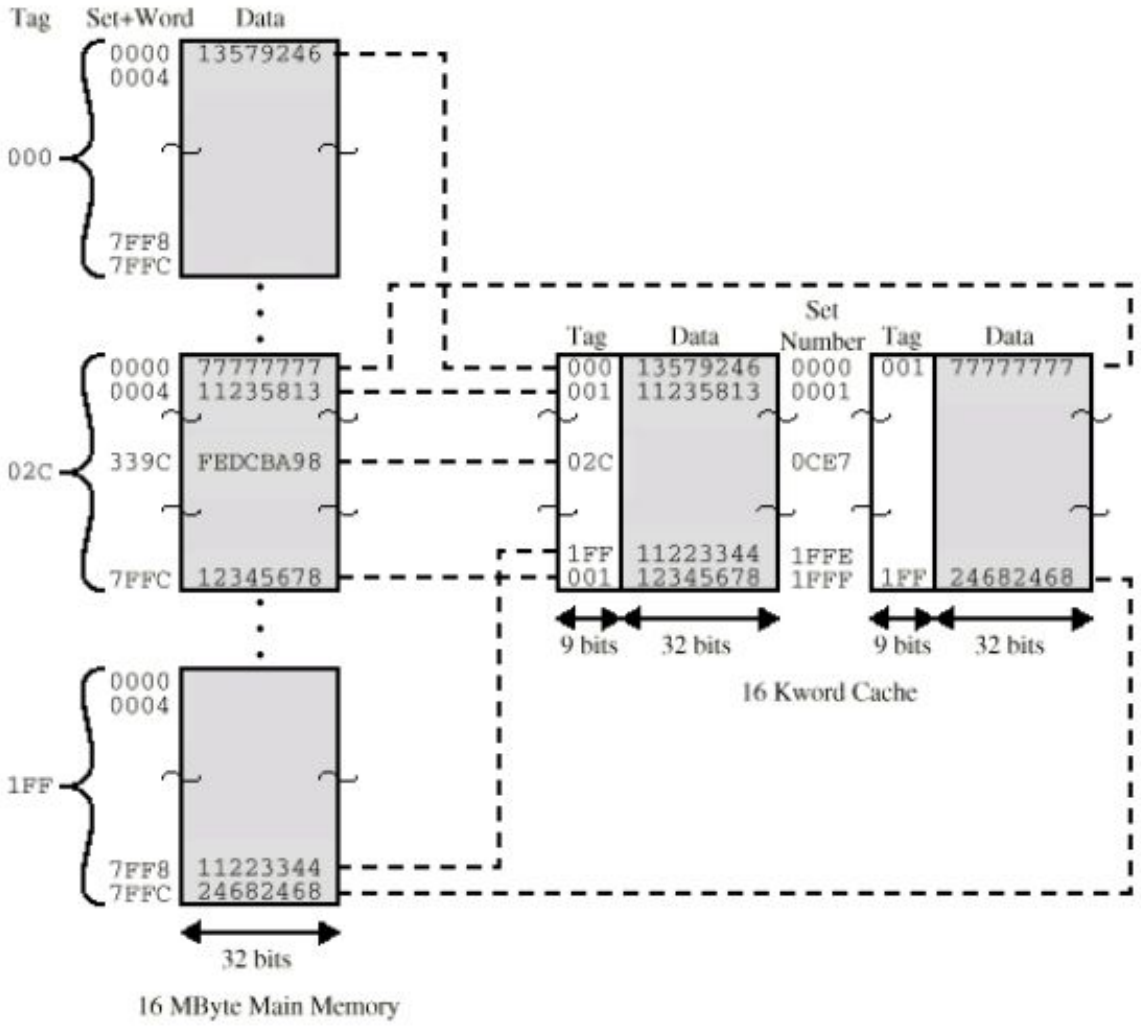
- Se usa el campo Set para determinar en cuál conjunto buscar
- Se compara con campo Tag para chequear hit
- En nuestro ejemplo, para dos vías:
  - 13 bits para el campo Set
  - El conjunto correspondiente a un bloque queda determinado mediante el número de bloque en memoria módulo 2^{13}
  - 000000, 00A000, 00B000, 00C000 … corresponden al mismo conjunto



Organización de la cache con correspondencia por conjuntos

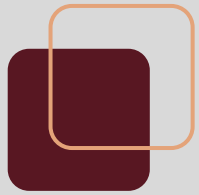


Ejemplo correspondencia "Two Way Set Associative"



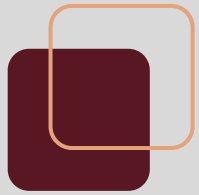
Tag : 9 bits
Set : 13 bits
Word : 2 bits

Algoritmos de sustitución
Correspondencia Directa



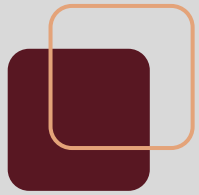
- No hay opciones
- Cada bloque corresponde a una línea
- Se debe reemplazar esa línea

Algoritmos de sustitución Correspondencia Totalmente Asociativa & “Set Associative”



- Implementado en hardware (velocidad)
- Least Recently Used (LRU)
  - Ej. en 2 way set associative
    - Un bit de USO determina cuál línea del conjunto debe ser reemplazada
  - First in first out (FIFO)
    - Reemplazar bloque “más viejo”
    - Round-robin, buffer circular
  - Least frequently used
    - Reemplazar bloque con menos hits
    - Contador?
  - Random

Algoritmos de sustitución (2/2)



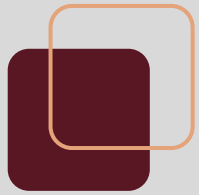
- Random
  - Fácil de implementar
- LRU
  - Difícil de implementar; frecuentemente aproximado
- FIFO
  - Se usa como aproximación de LRU
  - El efecto es menor (ver tabla); se acentúa con cachés pequeñas y de baja asociatividad

Algoritmos de sustitución (1/2)

	Associativity								
	Two-way			Four-way			Eight-way		
Size	LRU	Random	FIFO	LRU	Random	FIFO	LRU	Random	FIFO
16 KB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64 KB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256 KB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

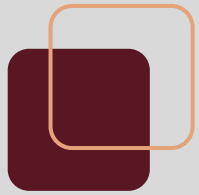
Figure C.4 Data cache misses per 1000 instructions comparing least-recently used, random, and first in, first out replacement for several sizes and associativities. There is little difference between LRU and random for the largest-size cache, with LRU outperforming the others for smaller caches. FIFO generally outperforms random in the smaller cache sizes. These data were collected for a block size of 64 bytes for the Alpha architecture using 10 SPEC2000 benchmarks. Five are from SPECint2000 (gap, gcc, gzip, mcf, and perl) and five are from SPECfp2000 (applu, art, equake, lucas, and swim). We will use this computer and these benchmarks in most figures in this appendix.

Políticas de escritura (1/2)



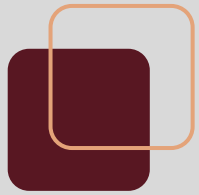
- Lectura del cache es más sencilla de manejar que la escritura
  - Cache de instrucciones más sencillo que cache de datos
- Escritura
  - ¿Cómo mantener la consistencia?

Políticas de escritura (2/2)



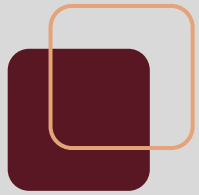
- Write Back: solo se escribe en caché. Se escribe en memoria cuando el bloque se debe reemplazar en un cache miss
  - Se necesita un "dirty bit" en cada bloque del caché
  - Reduce sensiblemente los requerimientos de ancho de banda de memoria
  - El control puede ser complejo
- Write Through: escribir en caché y memoria al mismo tiempo

Política “Write back”



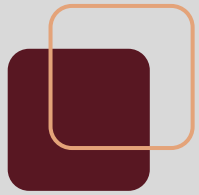
- Actualización se hace inicialmente sólo en la cache
- Set de “bit de actualización” cuando se hace una escritura
- Cuando se reemplaza el bloque, se escribe en memoria principal sólo si el “bit de actualización” está seteado
- Problemas
  - Sincronismo con otras caches
  - E/S debe acceder a memoria principal usando la caché
  - Cuello de botella potencial: Experiencia - 15% de referencias a memoria son escrituras

Política “Write through”



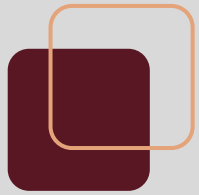
- Todas las escrituras se hacen en memoria y caché
- Múltiples CPUs pueden monitorizar el tráfico de memoria principal para actualizar caché local
- Mucho tráfico!

Causas de los “Cache Misses” (1/2)



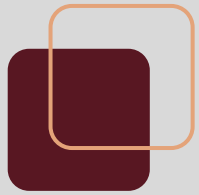
- Compulsivo (Obligatorio) (arranque, primera referencia):
  - No se puede hacer nada por evitarlo
  - Si se van a ejecutar “millones” de instrucciones, los Compulsory Misses son insignificantes.
- Conflicto (colisión):
  - Múltiples zonas de memoria mapeadas a la misma ubicación en caché.
  - Solución 1: incrementar tamaño de caché
  - Solución 2: incrementar la asociatividad !

Causas de los “Cache Misses” (2/2)



- Capacidad:
  - El caché no puede contener todos los bloques accedidos por el programa
  - Solución: incrementar el tamaño de la caché
- Coherencia (Invalidez): otros dispositivos actualizan la memoria (Procesadores, E/S)

Fin



¿Preguntas?