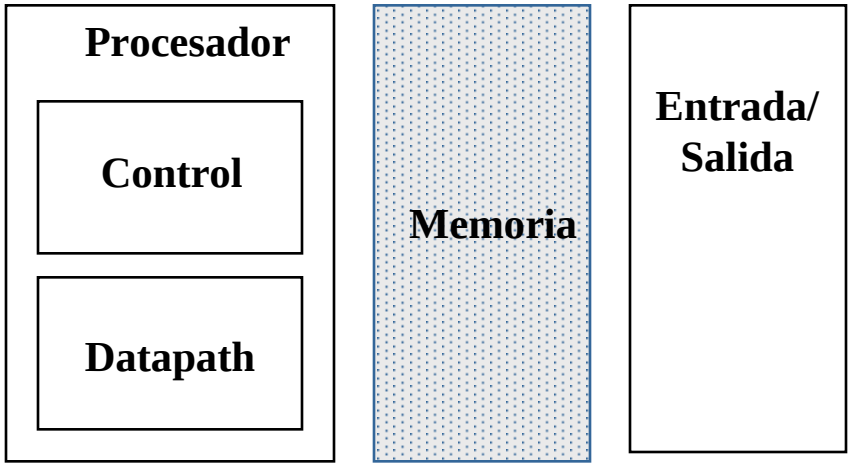
Arquitectura de Computadoras
Jerarquía de Memoria

Facultad de Ingeniería
Universidad de la República

Instituto de Computación
Curso 2011

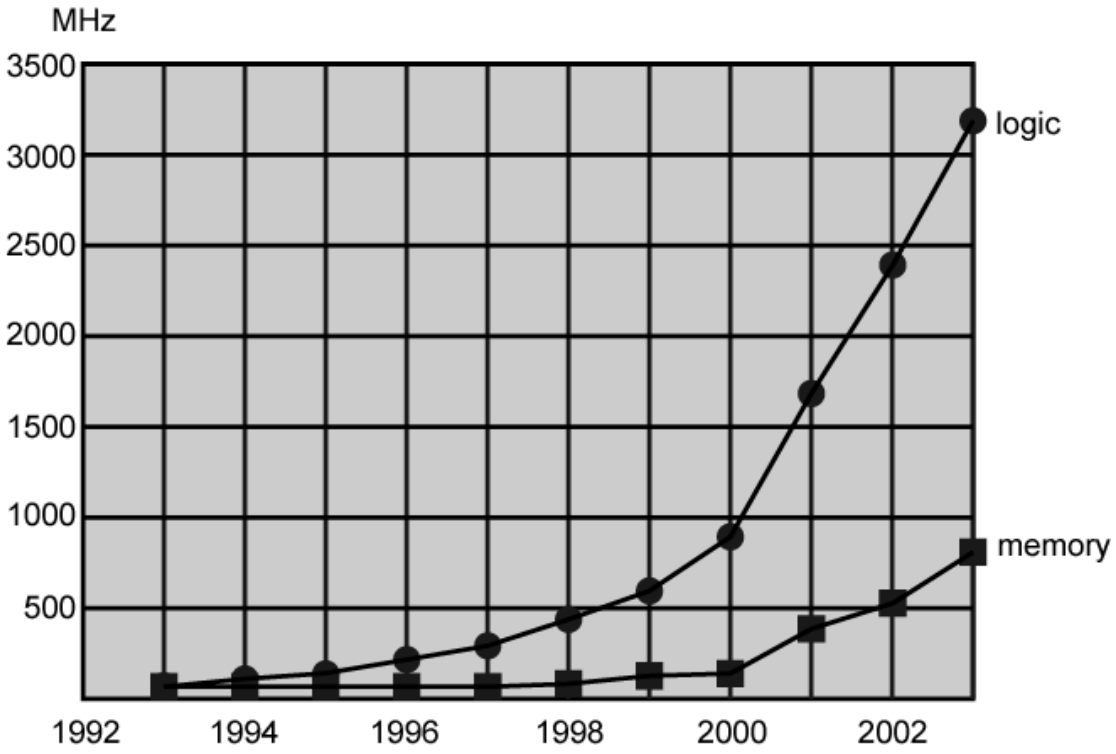
Recapitulando:
donde estamos?

- Componentes clásicos de un computador

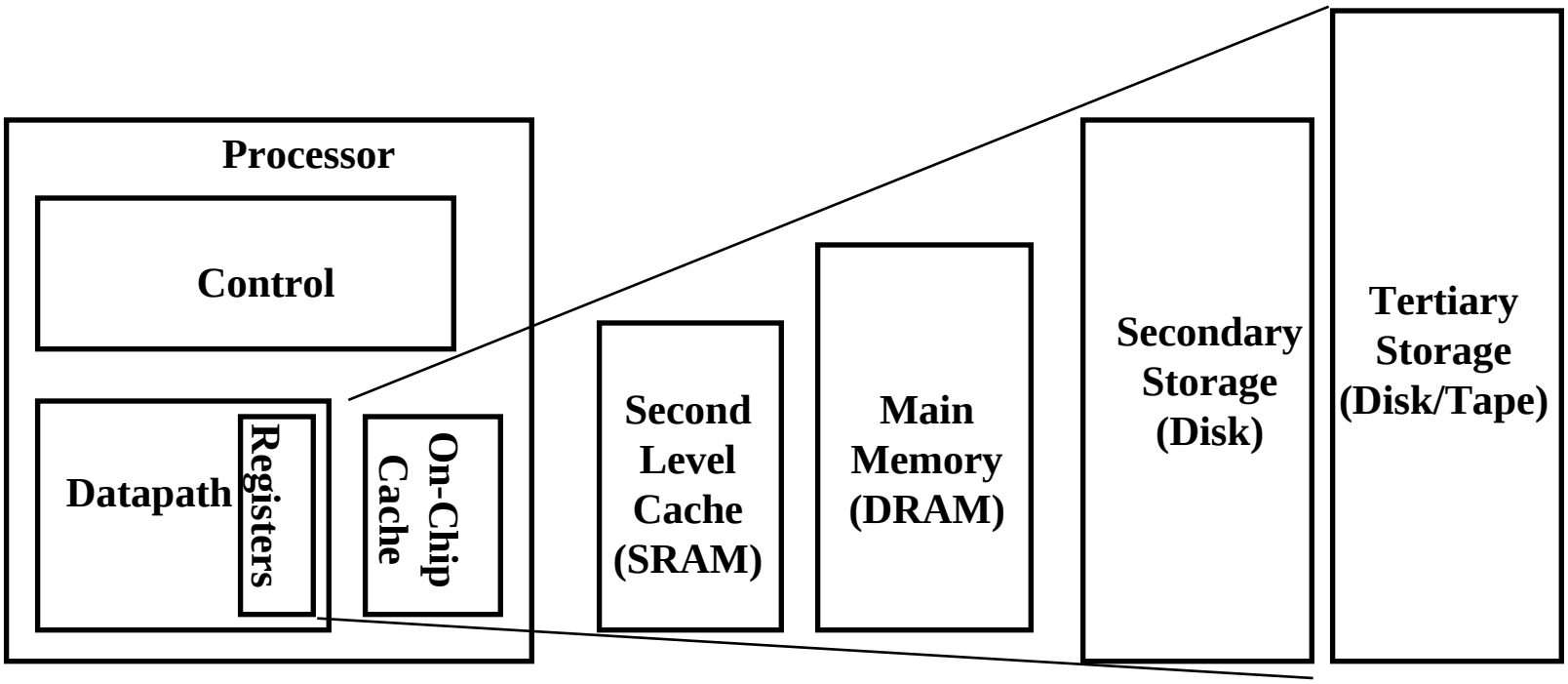


- Veremos otras ideas para mejorar el rendimiento
  - Localidad y jerarquía de memoria
  - Memorias SRAM, DRAM
  - Organización de la memoria
  - Caches, memoria virtual

Tendencia tecnológica



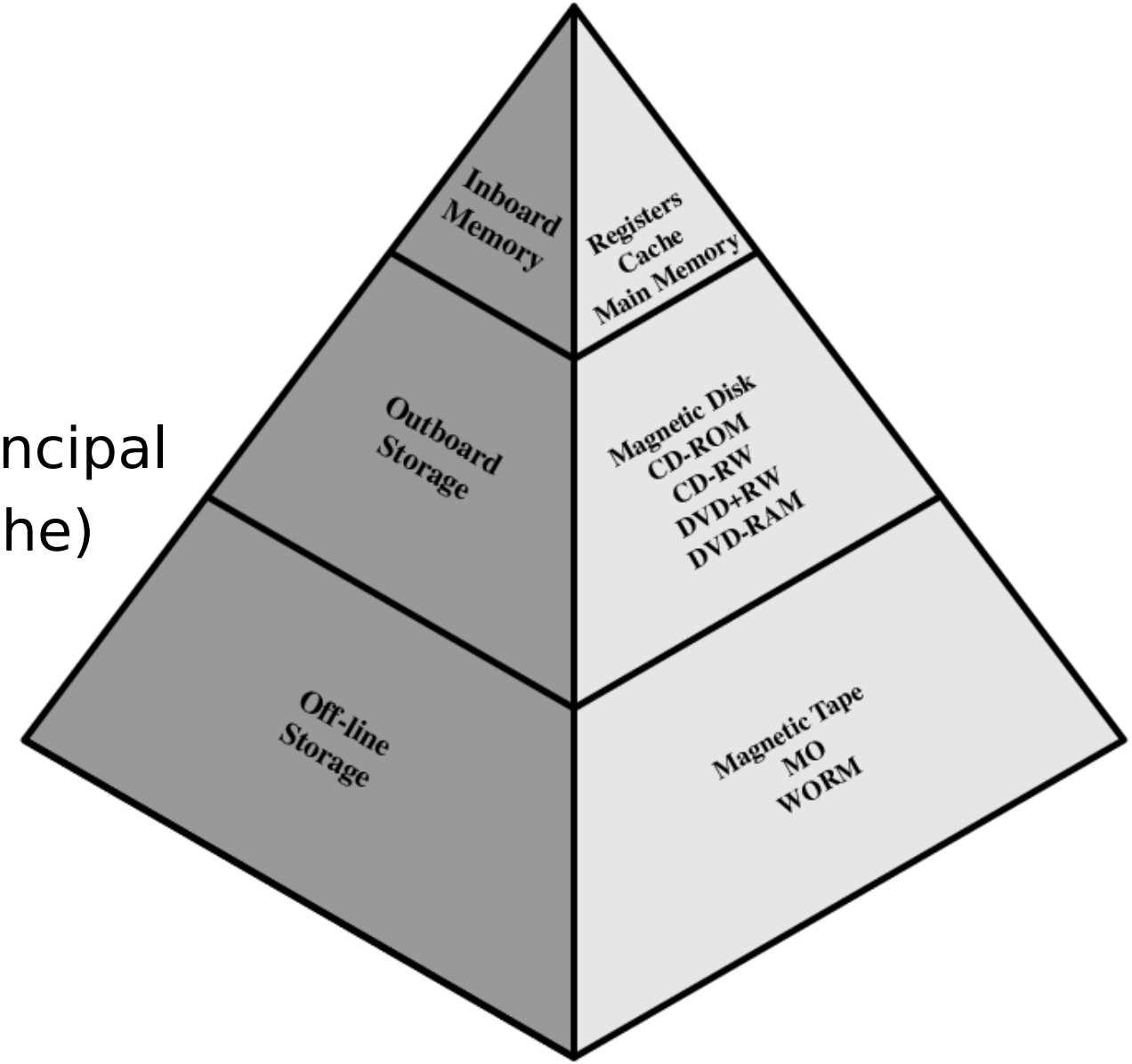
Jerarquía de memoria



Speed (ns):	1s	10s	100s	10,000,000s (10s ms)	10,000,000,000s (10s sec)
Size (bytes):	100s	Ks	Gs	100Gs	Ts

Jerarquía de memoria

- Registros
- Cache L1
- Cache L2
- Memoria Principal
- Disco (c/cache)
- Alm. óptico
- Cintas



Por qué funciona la jerarquía ?

- 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 los ítems con direcciones “cercanas”también sea 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

Como se maneja la jerarquía?

- Registros <-> Memoria
  - por el compilador (programador?)
- cache <-> memoria
  - por el hardware
- memoria <-> discos
  - por el hardware y el sistema operativo (memoria virtual)
  - por el programador (archivos)

Jerarquía de Memoria:
Terminología

- 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
- 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

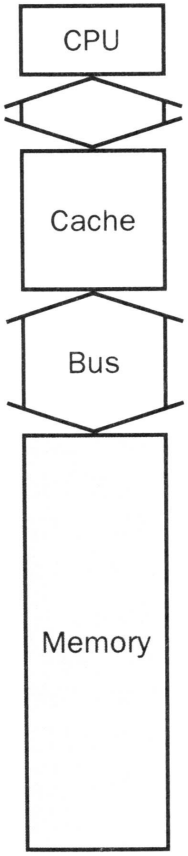
Tecnologías de Memoria:

- Accesso “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
- 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)

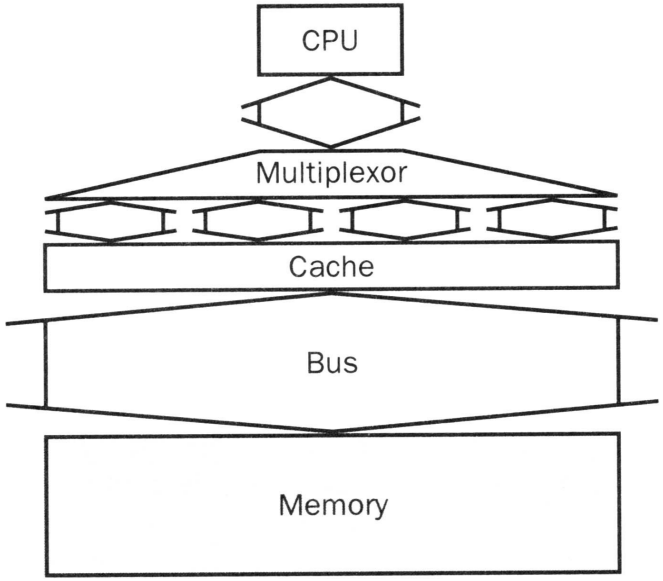
Algunas medidas de rendimiento de memoria

- 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 a la que se mueven los datos

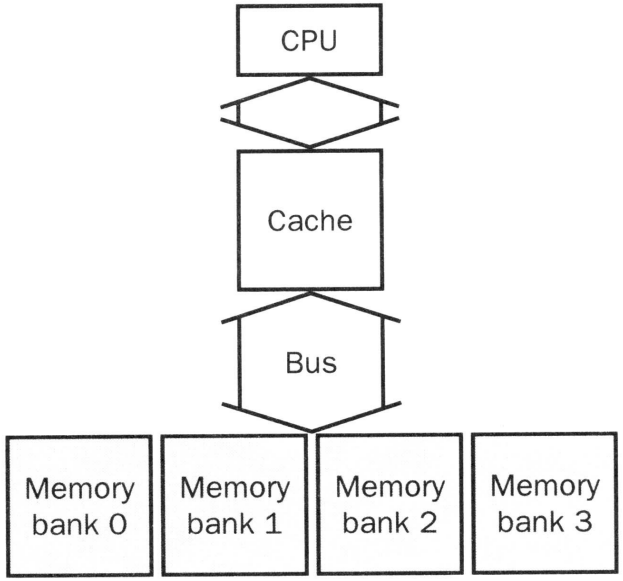
Memoria Principal: La organización puede mejorar el rendimiento



a. One-word-wide memory organization



b. Wide memory organization

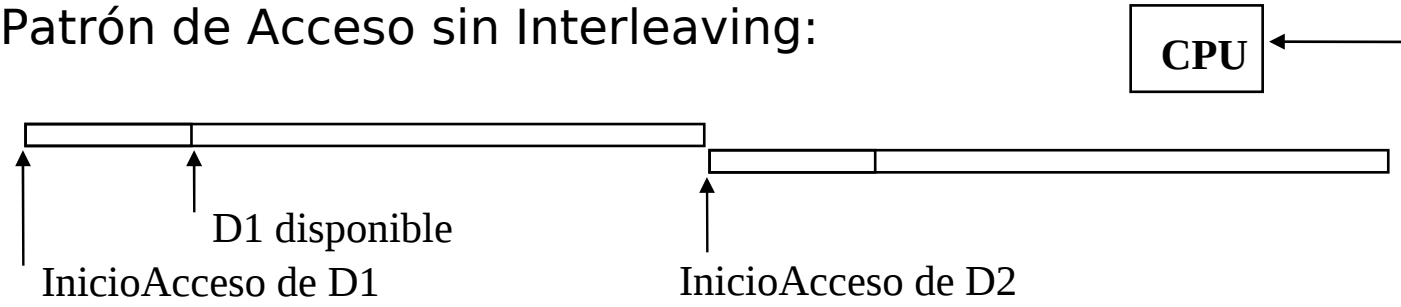


c. Interleaved memory organization

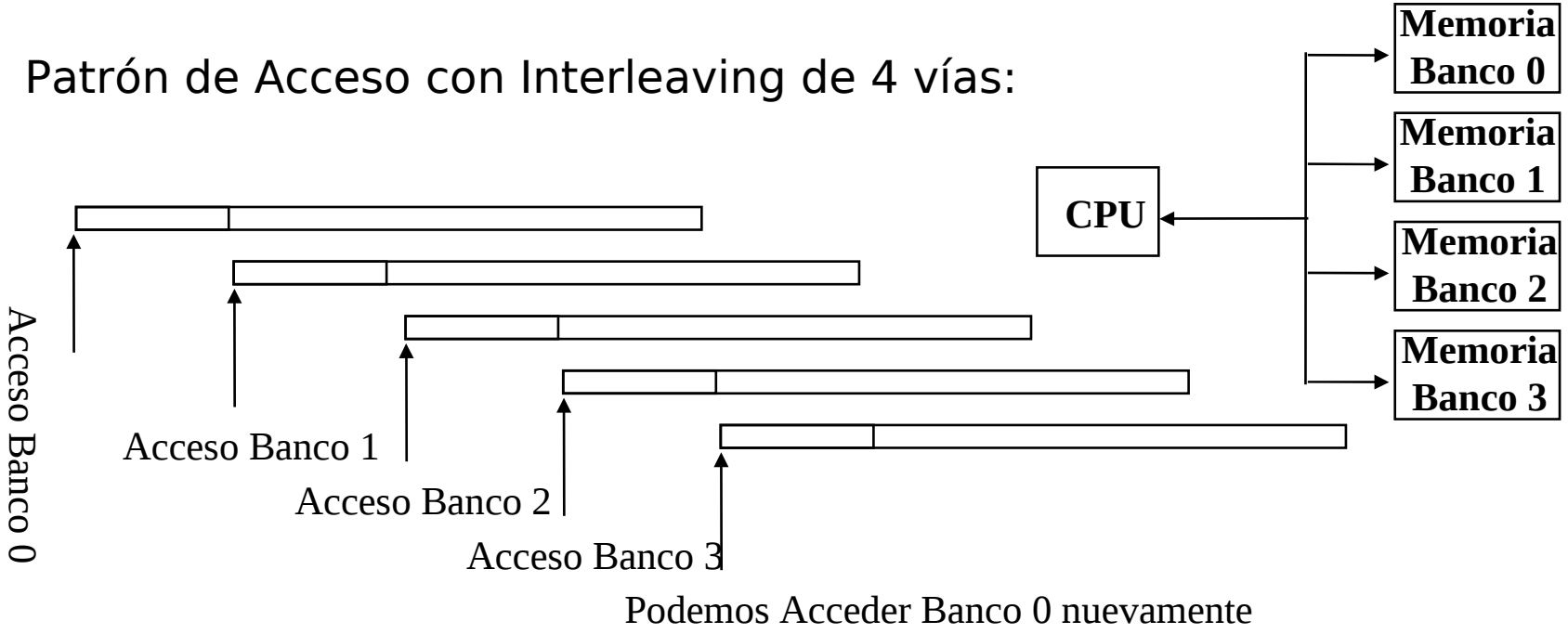
- Simple:
  - CPU, Cache, Bus, Memoria mismo ancho(32 bits)
- Ancho (Wide):
  - CPU/Mux 1 palabra; Mux/Cache, Bus, Memoria N palabras
- Entrelazado (Interleaved):
  - CPU, Cache, Bus 1 palabra: Memoria N Módulos; el ejemplo es de 4 módulos, word interleaved (estrelazado de palabra)

Entrelazado (Interleaving)

Patrón de Acceso sin Interleaving:

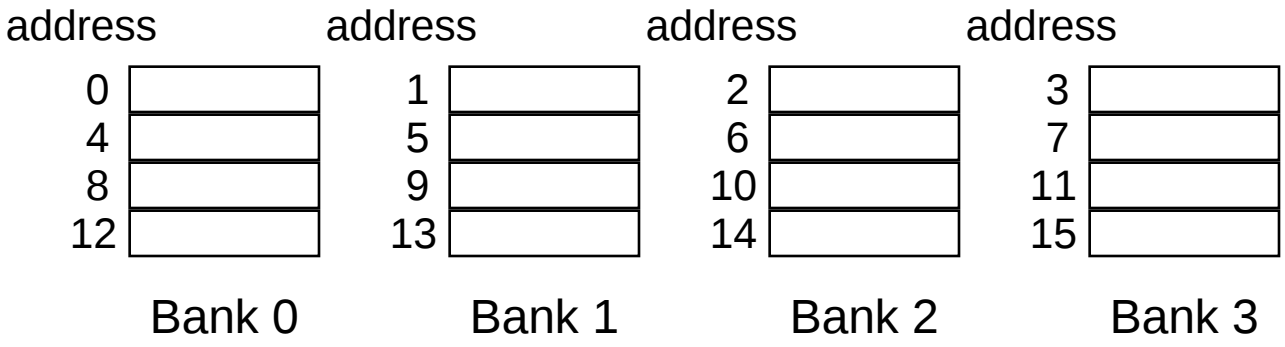


Patrón de Acceso con Interleaving de 4 vías:



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

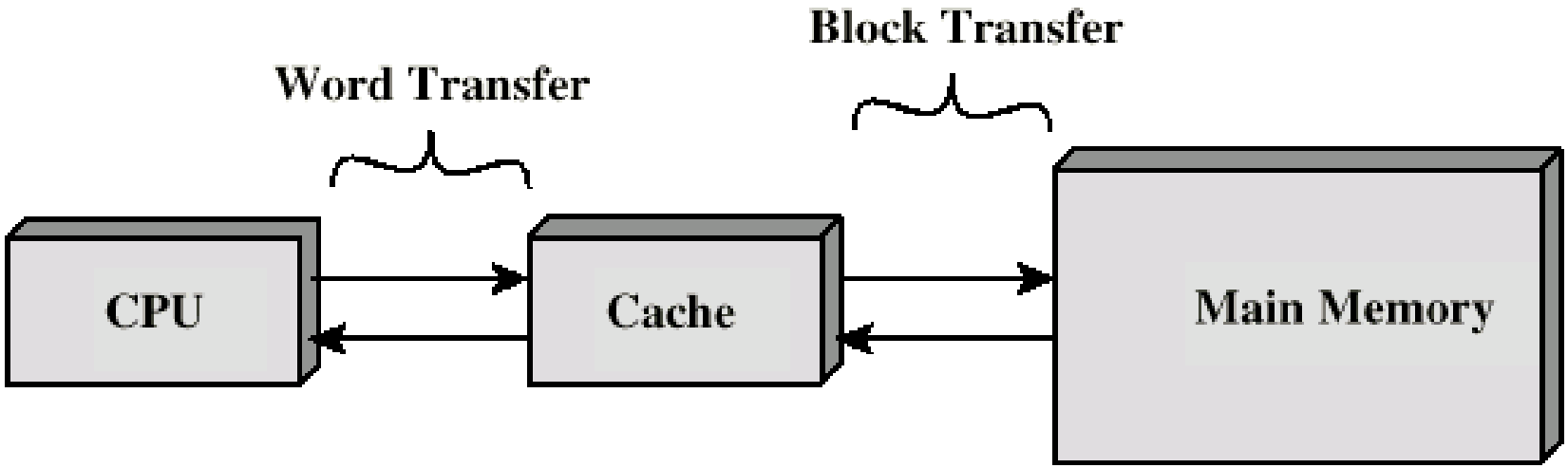


Resumen hasta el momento

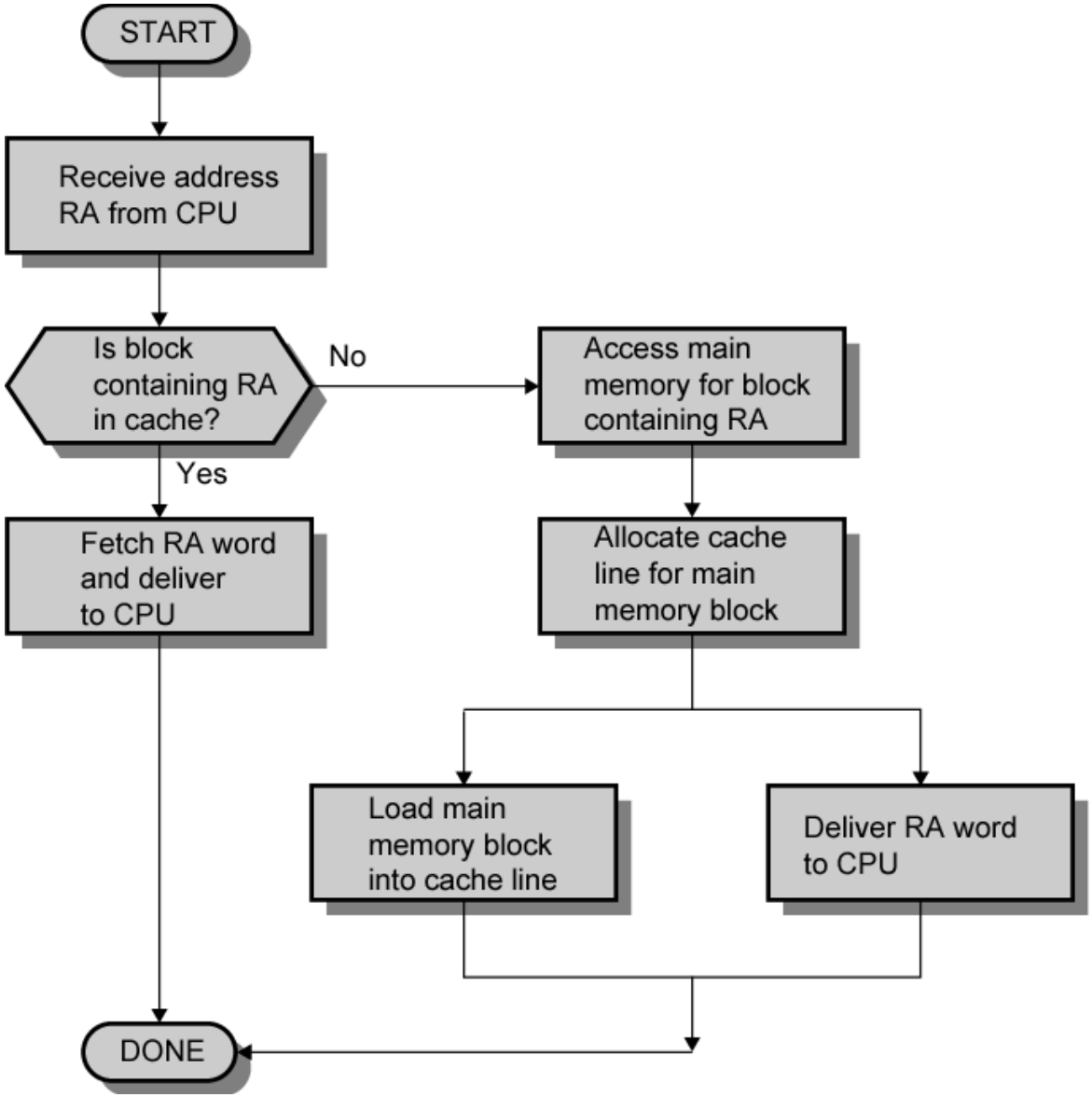
- Dos Tipos de Localidad:
  - Temporal
  - Espacial
- Tomando ventaja del principio de localidad:
  - Construir un sistema con mucha memoria de tecnología baratas
  - Proveer acceso a la velocidad ofrecida por las tecnologías más rápidas
- La DRAM es lenta, pero de alta densidad y barata:
  - Presenta al usuario un sistema de memoria GRANDE.
- La SRAM es rápida, pero de baja densidad y cara:
  - Presentar al usuario un sistema de memoria RÁPIDO.
- El Arte del diseño de un Sistema de Memoria consiste en Optimizar la organización para minimizar el tiempo de acceso promedio para las cargas de trabajo típicas

Cache

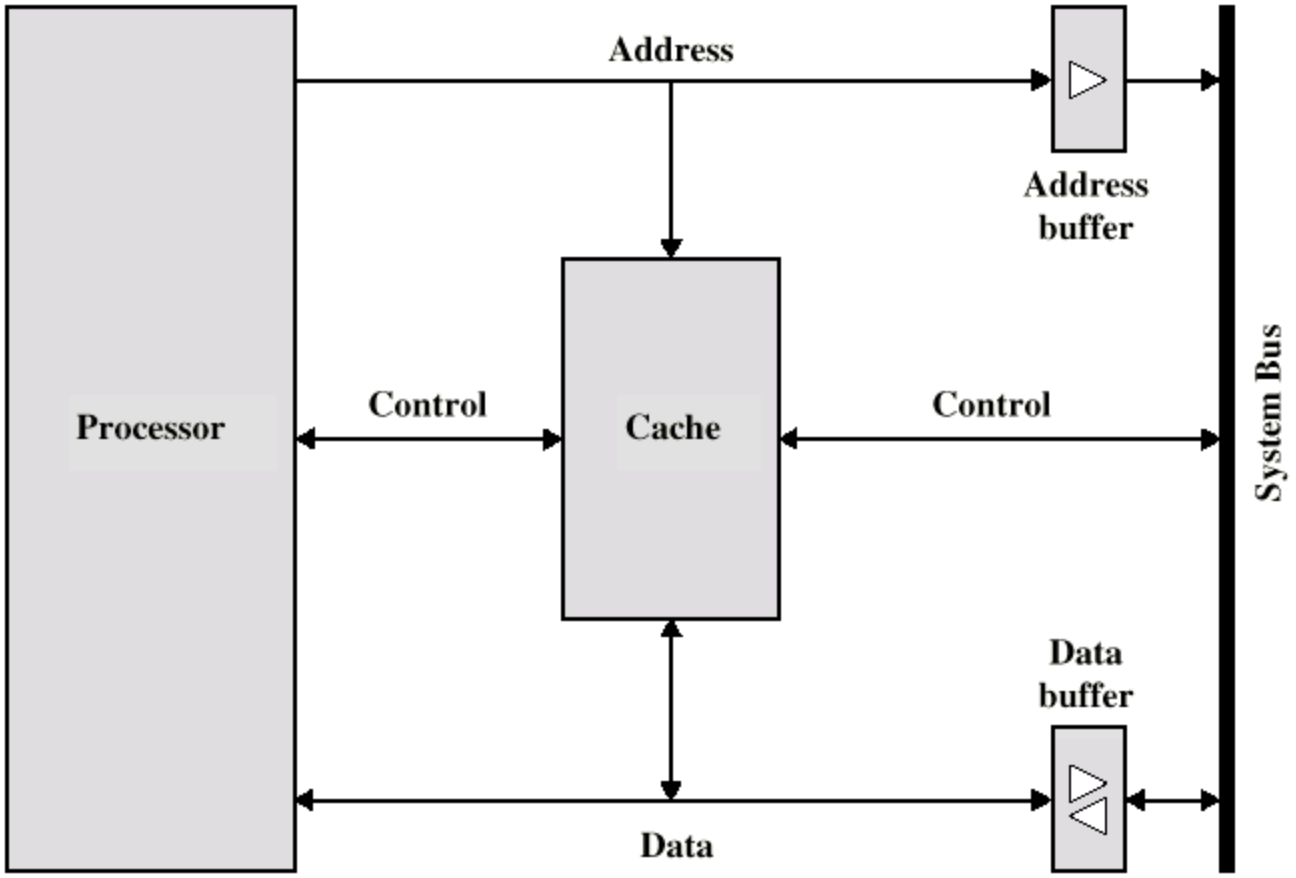
- Memoria rápida, escasa
- Se instala entre la memoria principal y la CPU
- Puede esta en el chip de la CPU (L1) o módulo externo (L2)
- La memoria principal se accede por *bloques* de K palabras (en general K > 1). Cuando se lee un bloque, se almacena en una *línea* de cache.



Operativa de la cache: lectura



Organización típica del cache



Diseño de la Cache

- Tamaño
  - Compromiso Costo/Velocidad
    - Más cache, más velocidad (hasta cierto punto)
- Función de correspondencia (Mapping Function)
  - Entre bloques de memoria principal y líneas del cache
- Algoritmo de sustitución
  - ¿Qué bloque se desecha cuando se lee un bloque nuevo?
- Política de escritura
  - ¿Hubo modificaciones?
- Tamaño del bloque
  - Para un tamaño de cache dado: ¿más líneas o líneas más grandes?
- Cantidad de caches
  - ¿Cache unificada o caches separadas para datos e instrucciones?

Función de correspondencia

- Para todos los casos consideraremos un ejemplo:
  - Cache de 64 KBytes
  - Bloques de 4 bytes
    - cache de 16K (2^{14}) líneas de 4 bytes cada una
  - 16MBytes de memoria principal
  - 24 bits de direcciones
    - (2^{24}=16 MB)

Correspondencia Directa

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

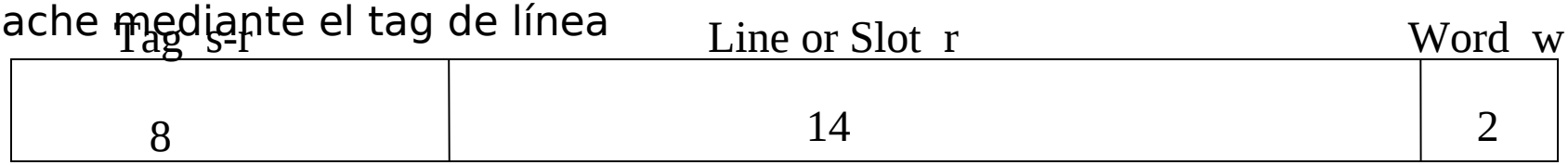
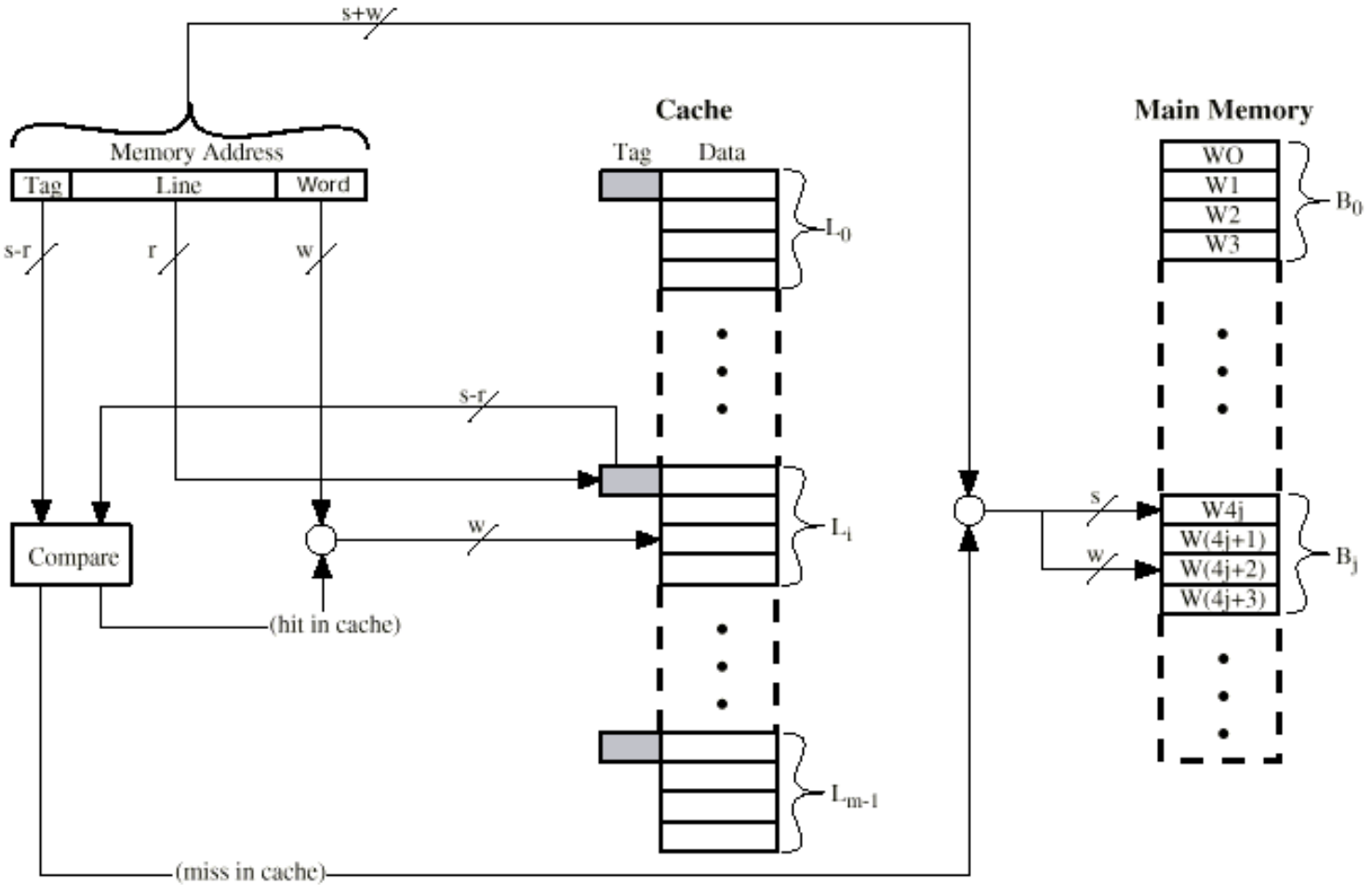


Tabla de líneas de la cache con Correspondencia Directa

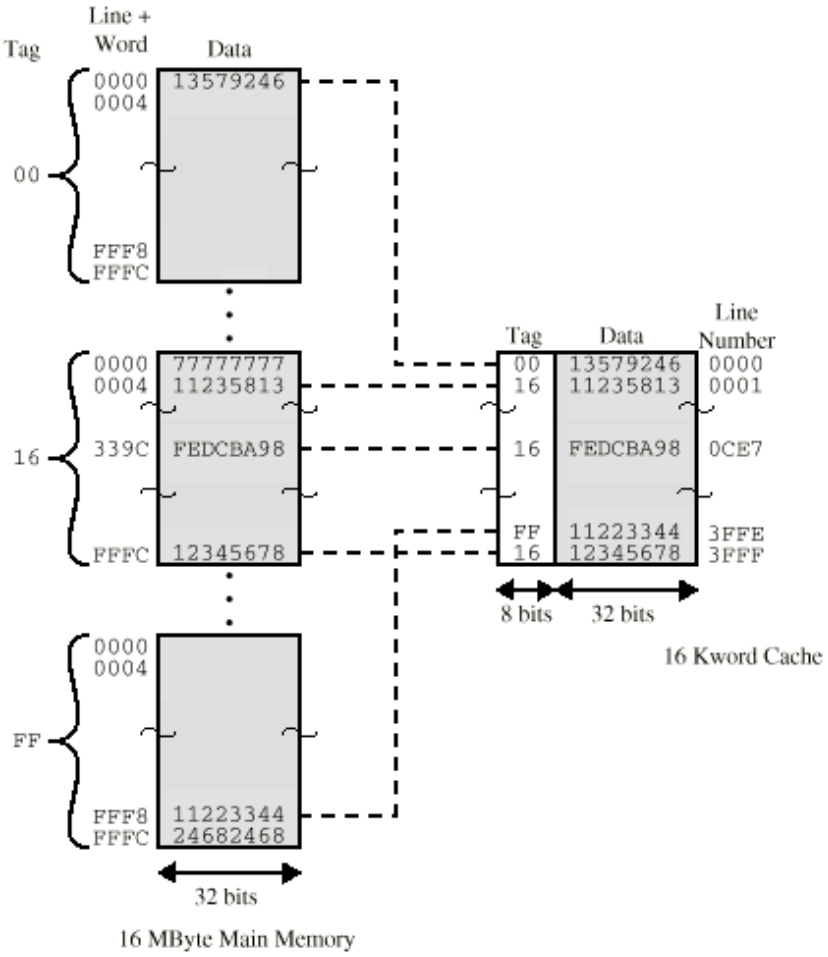
- Correspondencia:
  - i=j modulo m
    - i línea del cache
    - j número de bloque de memoria principal
    - m=2^r (cantidad de líneas del cache)

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

Organización de la cache con correspondencia directa



Ejemplo correspondencia directa



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

Resumen correspondencia directa

- Direcciones = (s + w) bits
- Unidades direccionables = 2^{s+w} words o bytes
- Tamaño del bloque = tamaño de línea = 2^{w} words o bytes
- Cantidad de bloques en memoria principal = $2^{s+w}/2^{w} = 2^{s}$
- Cantidad de líneas de la cache = m = 2^{r}
- Tamaño del tag = (s – r) bits

- Pros & contras
  - Simple
  - Barato
  - Localización fija de cada bloque
    - Si un programa accede a 2 bloques que se corresponden a una misma línea repetidamente, genera numerosos “cache misses”

Correspondencia Asociativa

- Un bloque de memoria principal se puede cargar en cualquier línea de la cache
- 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

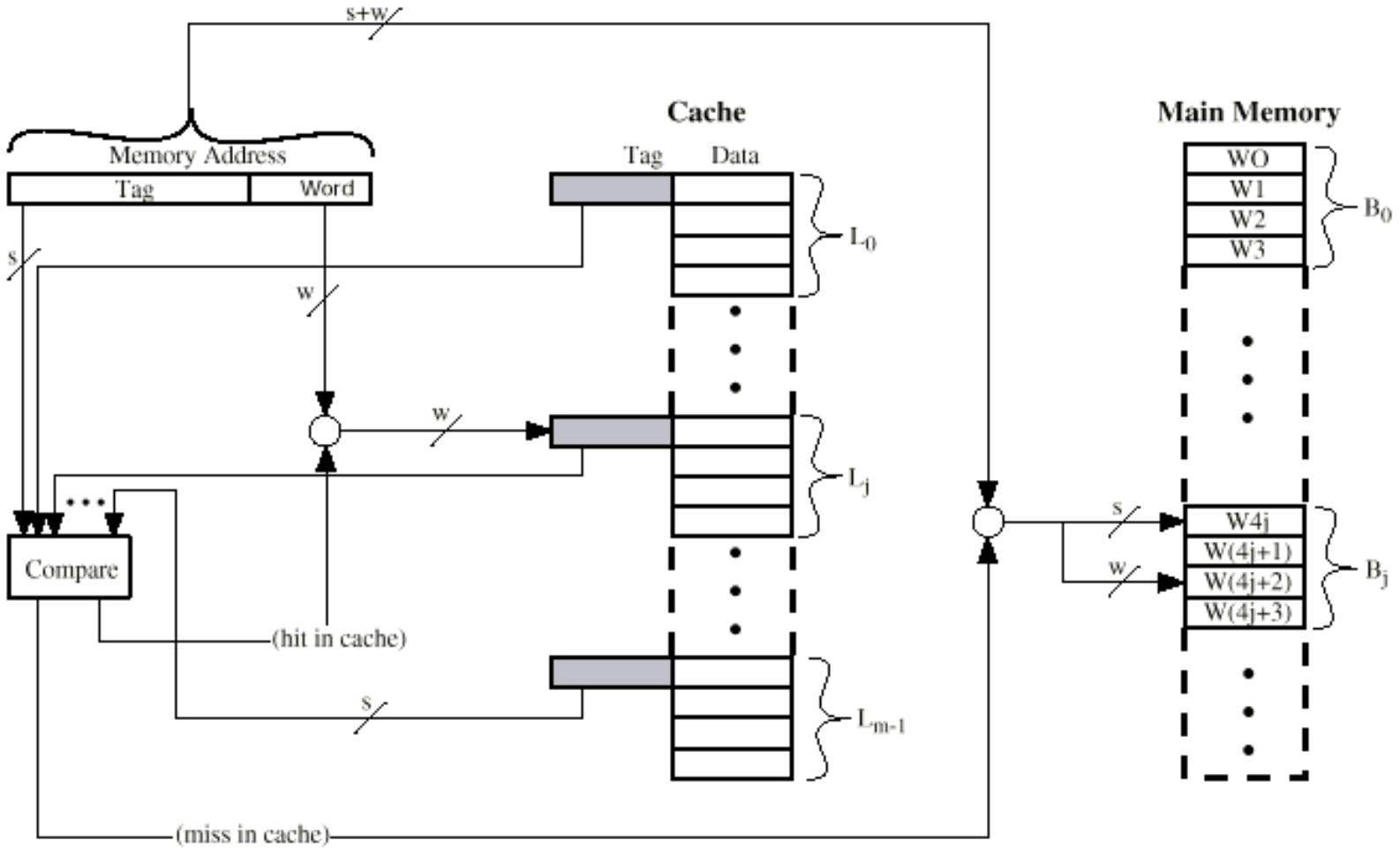
Estructura de direccionamiento correspondencia asociativa

En nuestro ejemplo:

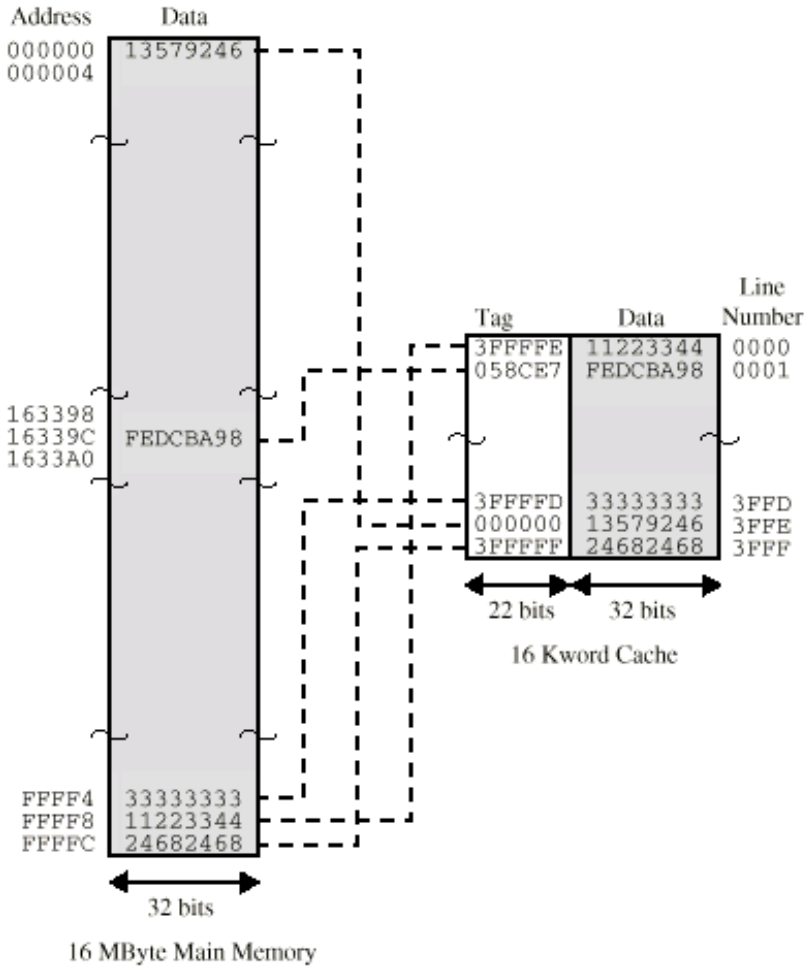
- Tag de 22 bits almacenada con cada bloque de datos de 32 bits
- Se compara el campo tag con la tag guardada en la cache para chequear un hit
- 2 bits menos significativos de la dirección identifican la palabra

Tag 22 bit	Word 2 bit

Organización de la cache con correspondencia asociativa (fully associative)



Ejemplo correspondencia asociativa



Tag : 22 bits
Word : 2 bits

Resumen correspondencia asociativa

- Direcciones = (s + w) bits
- Unidades direccionables = 2^{s+w} words o bytes
- Tamaño bloque = tamaño de línea = 2^{w} words o bytes
- Cantidad de bloques en memoria principal = $2^{s+w}/2^{w} = 2^{s}$
- Cantidad de líneas de cache = indeterminado
- Tamaño del tag = s bits

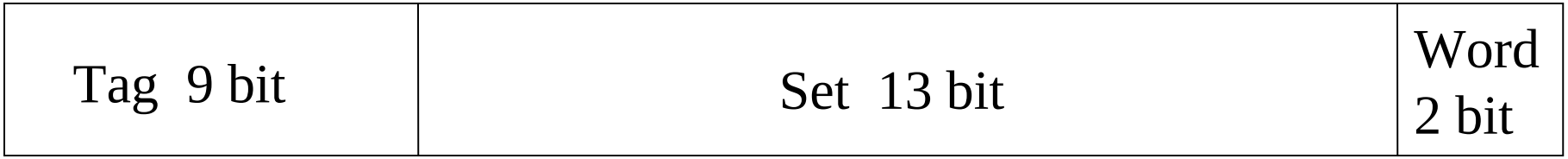
Correspondencia asociativa de conjuntos
(Set Associative)

- Cache 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

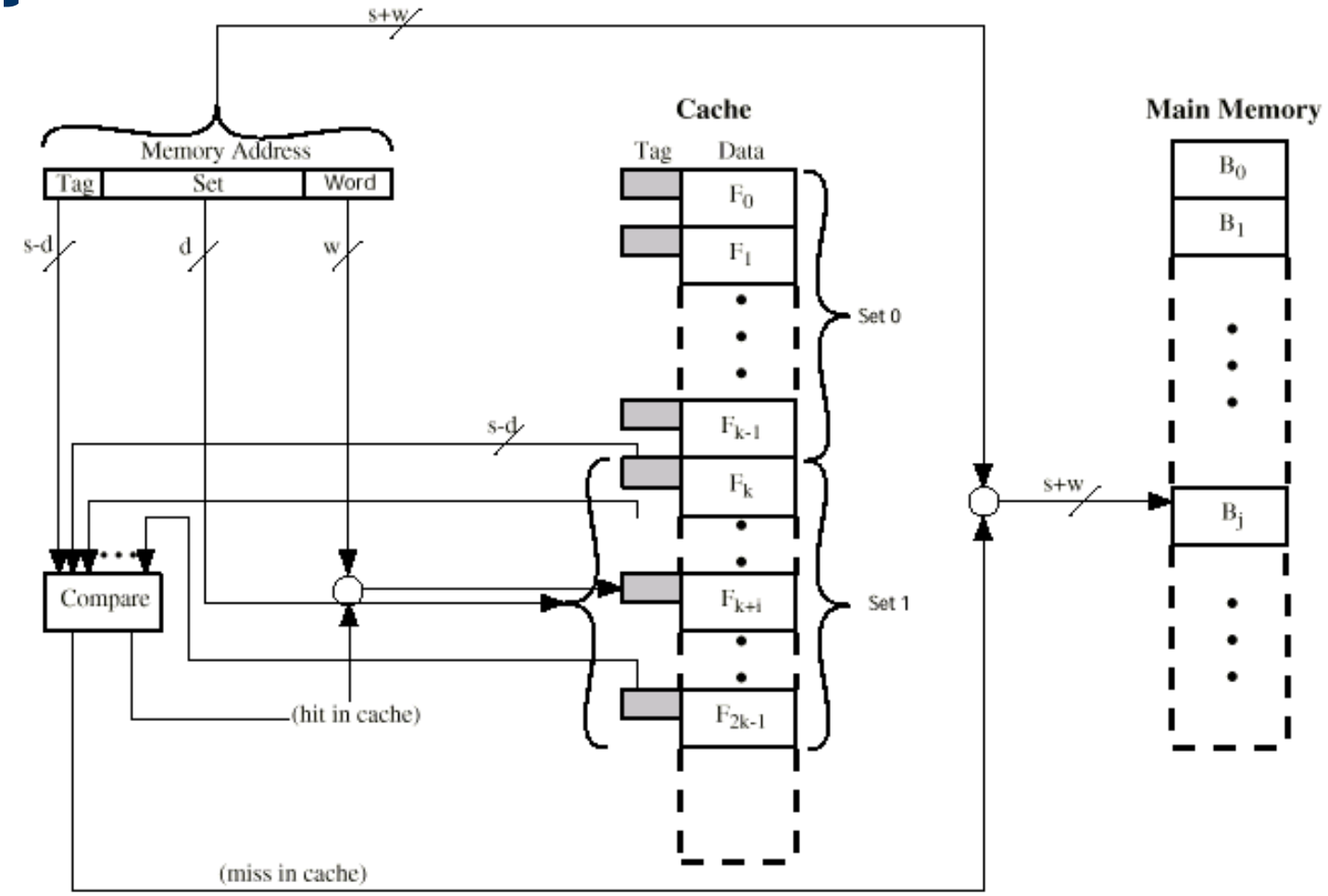
Estructura de direccionamiento “Set Associative Mapping”

- Se usa el campo Set para determinar en cuál conjunto buscar
- Se compara con campo Tag para chequear hit

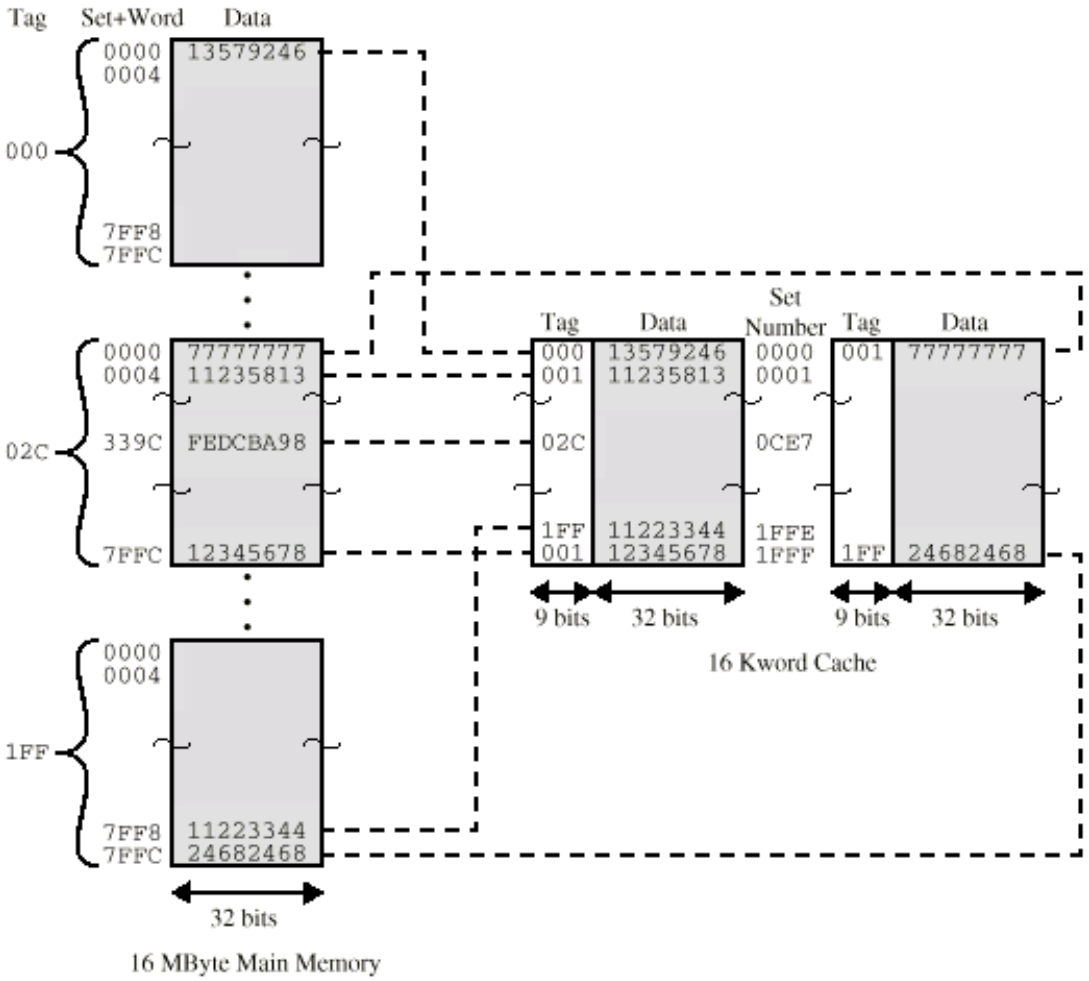
- En nuestro ejemplo, para dos vías:
  - 13 bits para el campo Set
  - El conjunto correspondiente a un bloque queda determinado mediante el número de bloque en memoria módulo 2^{13}
  - 000000, 00A000, 00B000, 00C000 … corresponden al mismo conjunto



Organización de la cache con correspondencia asociativa de conjuntos



Ejemplo correspondencia “Two Way Set Associative”



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

Resumen correspondencia “Set Associative”

- Dirección = (s + w) bits
- Unidades direccionables = 2^{s+w} words o bytes
- Tamaño bloque = tamaño línea = 2^{w} words o bytes
- Cantidad de bloques en memoria principal = 2^{d}
- Cantidad de líneas en conjunto = k
- Cantidad de conjuntos = v = 2^{d}
- Cantidad de líneas en cache = kv = k * 2^{d}
- Tamaño del tag = (s – d) bits

Algoritmos de sustitución (1/2)
Correspondencia Directa

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

Algoritmos de sustitución (2/2)
Correspondencia 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)
  - Reeemplazar bloque “más viejo”
    - Round-robin, buffer circular
- Least frequently used
  - Reemplazar bloque con menos hits
    - Contador?
- Random

Política de escritura

- No se debe sobreescribir un bloque de cache a menos que esté actualizado en memoria principal!
- Posibles conflictos
  - En sistemas con múltiples CPUs, estos pueden tener caches individuales
  - E/S puede acceder a memoria directamente

Política “Write through”

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

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 memoria principal usando la cache
    - Cuello de botella potencial
- Experiencia
  - 15% de referencias a memoria son escrituras

Resumen:
Causas de los “Cache Misses”

- 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 cache.
  - Solución 1: incrementar tamaño de cache
  - Solución 2: incrementar la asociatividad
- Capacidad:
  - El cache no puede contener todos los bloques accedidos por el programa
  - Solución: incrementar el tamaño de la cache
- Coherencia (Invalidez): otros dispositivos actualizan la memoria (Procesadores, E/S)

Preguntas?