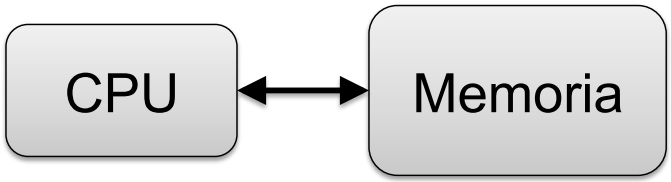
Tema 1. Jerarquía de memoria: Conceptos básicos

Organización de Computadores

LUIS ENRIQUE MORENO LORENTE
RAÚL PÉRULA MARTÍNEZ
ALBERTO BRUNETE GONZALEZ
DOMINGO MIGUEL GUINEA GARCIA ALEGRE
CESAR AUGUSTO ARISMENDI GUTIERREZ
JOSÉ CARLOS CASTILLO MONTOYA

Departamento de Ingeniería de Sistemas y Automática

CUELLO DE BOTELLA CPU-MEMORIA



- Las prestaciones de los computadores de alta velocidad suelen estar limitadas por el ancho de banda y la latencia de la memoria
  - Latencia (Latency) (tiempo para un único acceso)
    - Tiempo de acceso a memoria >> tiempo de ciclo del procesador
  - Ancho de banda (Bandwidth) (número de accesos por unidad de tiempo)
    - Si m es la fracción de instrucciones que accede a la memoria ,
      - 1+m referencias a memoria / instrucción
      - CPI = 1 requiere 1+m refs memoria / ciclo

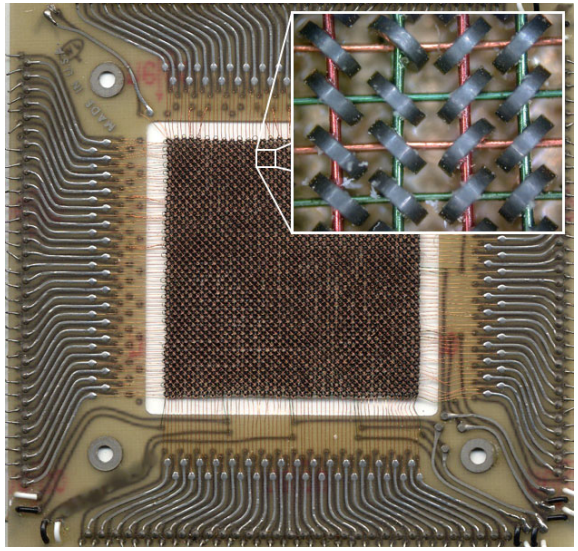
MEMORIA

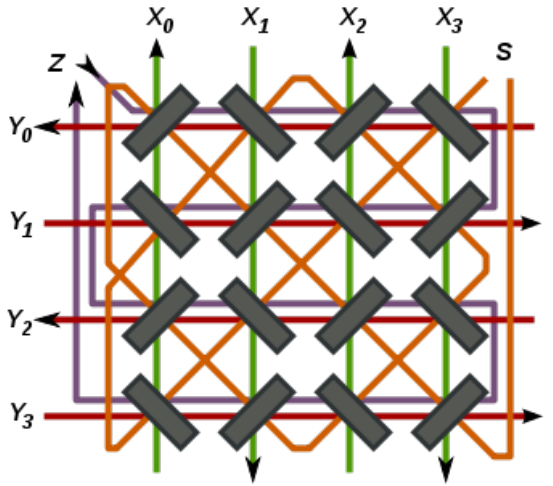
- Memoria principal
  - Memorias de ferritas (o memoria de toros)
    - Empiezan a usarse en los primeros computadores (años 40)
    - Producción manual.
  - Memoria de semiconductores
    - Comienzan a usarse en los años 70.
    - RAM, Random access memory
    - DRAM, Dynamic RAM
    - SRAM, Static RAM

CORE MEMORY/MEMORIA DE FERRITAS

- Core memory/memoria de ferritas fue la primera memoria principal fiable y de gran capacidad.
  - Inventada por Forrester a finales de los 40s en el MIT (Whirlwind project)
  - Los bits se almacenaban como una polarización magnética de unos pequeños nucleos de ferrita enhebrados en una red de hilos de 2 dimensiones
  - Pulsos coincidentes en los hilos X e Y escribían y leían el estado original (lecturas destructivas)
  - Almacenamiento robusto, no-volátil
  - Utilizado en computadores para el espacio hasta muy recientemente
  - Los nucleos se enhebraban manualmente (25 billones al año fue el pico de producción)
  - Tiempo de acceso ~ 1µs

X e Y son las direcciones, S es para inhibir, Z es para leer

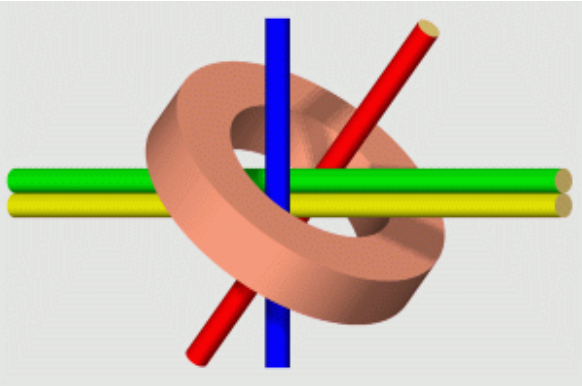




Imagenes de Wikipedia,2014

CORE MEMORY/MEMORIA DE FERRITAS

- Y de aquí viene el escudo de la profesión…



MEMORIAS DE SEMICONDUCTORES

- Las memorias de semiconductores empezaron a ser competitivas a comienzo de la década de 1970s
  - Intel diseño la primera y comenzó a explotar el mercado de las memorias de semiconductores
- La primera DRAM (dynamic RAM) comercial fue la Intel 1103
  - 1Kbit de capacidad en un único chip
  - Se carga la información en un condensador que se usa para mantener el valor.
- Las memorias de semiconductores reemplazaron a las de ferritas muy rápidamente en los 1970s
- Types
  - RAM (Random access memory)
  - ROM (Read-only memory): EPROM, EEPROM, etc.
  - Flash

RANDOM ACCESS MEMORY (RAM)

- Dynamic Random Access Memory (DRAM) (10 – 50 ns)
  - Alta densidad, bajo consumo, barata pero lenta
  - Dinámica, ya que los datos deben ser "refrescados" regularmente
  - El contenido se pierde al apagar el equipo
- Static Random Access Memory (SRAM) (0.5 - 2.5 ns)
  - Menor densidad (alrededor de 1/10 de la DRAM), mayor coste
  - Estática ya que los datos se mantienen sin necesidad de refresco mientras el equipo está encendido
  - Tiempo de acceso más rápido, a menudo de 2 a 10 veces más rápida que la DRAM
- Flash Memory (>60ns)
  - Mantiene los datos sin alimentación del equipo
  - Los datos se escriben en bloques, es generalmente lenta
  - Barata

DYNAMIC RAM DE 1 TRANSISTOR

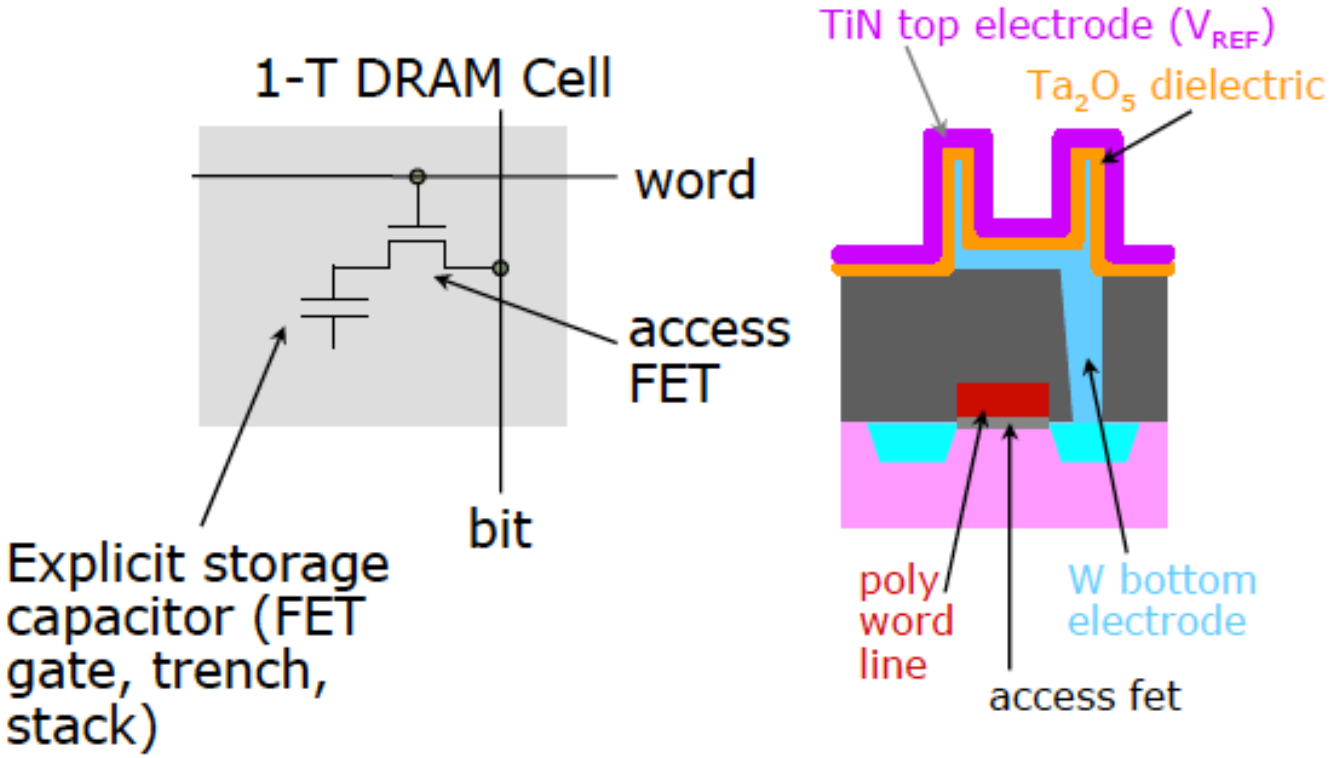
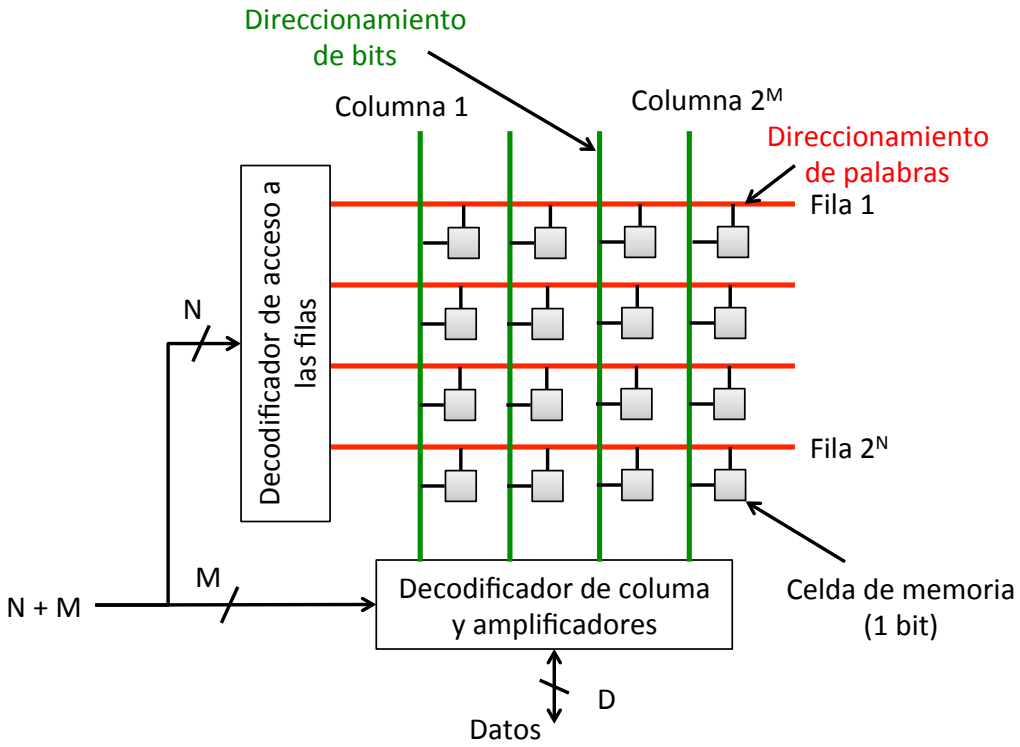


Imagen de Emer, 2005

DRAM: ARQUITECTURA

- Los bits se almacenan en arrays 2-dimensionales en el chip.
- Los chips actuales tienen 4 bancos lógicos por cada chip.
  - Cada banco lógico es implementado mediante muchos arrays de menor tamaño.



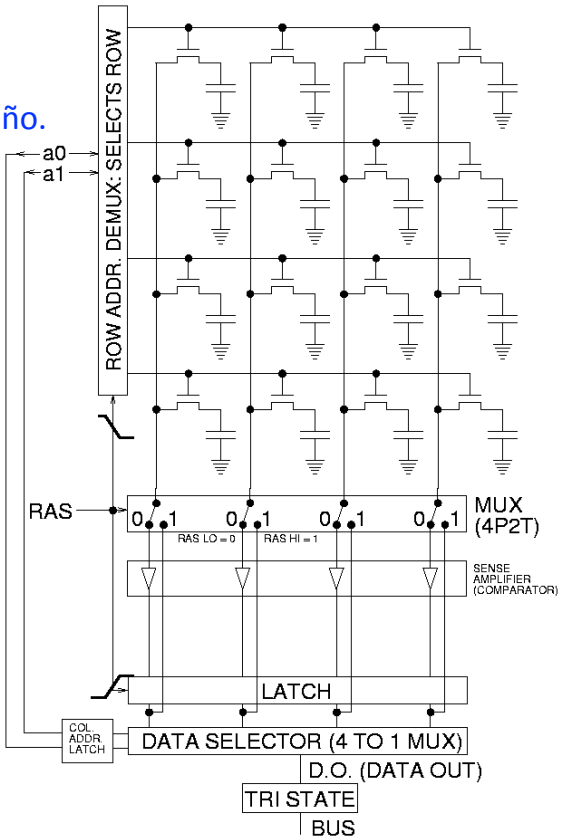


Imagen de Wikipedia, 2014b

DRAM: MODO DE OPERACIÓN

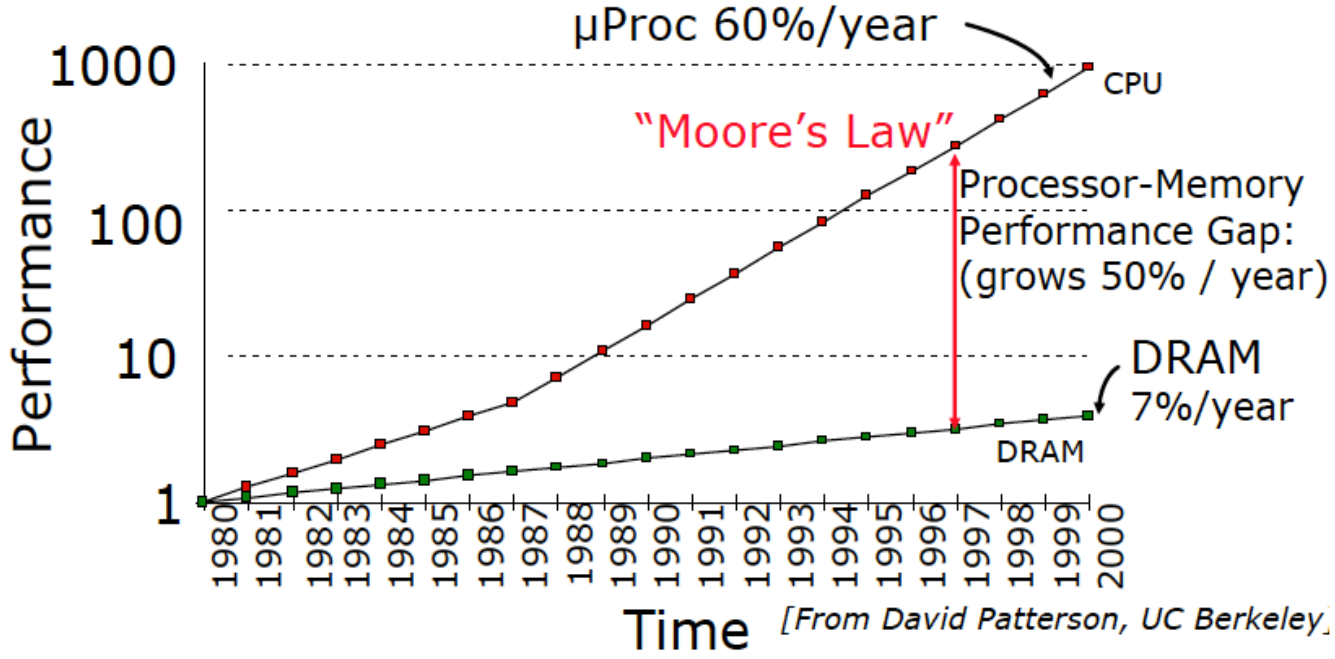
Pasos en una operación read/write sobre un banco de memoria.

1. Row access (RAS)
   - Decodif. de la dirección de fila, enable dirección de fila (a menudo multiple Kb por fila)
   - Las bitlines comparten la carga con la celda de memoria.
   - Pequeños cambios en el voltaje son detectados por amplificadores de medida con latch para toda la fila de bits, y amplificados para recargar las celdas de memoria.
2. Column access (CAS)
   - Decodificación de la dirección de columna para seleccionar un pequeño numero de latches amplificadores de medida (4, 8, 16, or 32 bits dependiendo del DRAM).
   - En la lectura, se envían los bits latched a los pins del chip.
   - En la escritura, los cambios medidos por los latches amplificadores se usan para cargar las celdas de memoria al valor deseado.
   - Pueden realizar múltiples accesos a columna en la misma fila sin otro acceso a fila(burst mode).
3. Precharge
   - Líneas de carga de bit a valor conocido, requerido antes de un nuevo acceso a fila.

- Cada paso tiene una latencia de unos 20ns en las DRAMs modernas.
- Varios DRAM standards (DDR, RDRAM) tienen diferentes formas de codificar las señales para la transmisión a la DRAM, pero la arquitectura suele ser la misma.

MOTIVACIÓN: GAP DE LATENCIA ENTRE CPU-DRAM

- El gap existente entre las prestaciones de memorias y procesadores.
- En 1980 ningún microprocesador llevaba cache.
- En 1995 2-level cache, 60% de los transistores (7.2 millones) del proc. Alpha 21164.
- Un procesador superescalar de 4-issues puede ejecutar 800 instr durante un cache miss!



LEY DE LITTLE

- The long-term average number of customers in a stable system L is equal to the long-term average effective arrival rate, λ, multiplied by the average time a customer spends in the system, W;

$$L = \lambda W$$

- Throughput (T) = Number in Flight (N) / Latency (L)

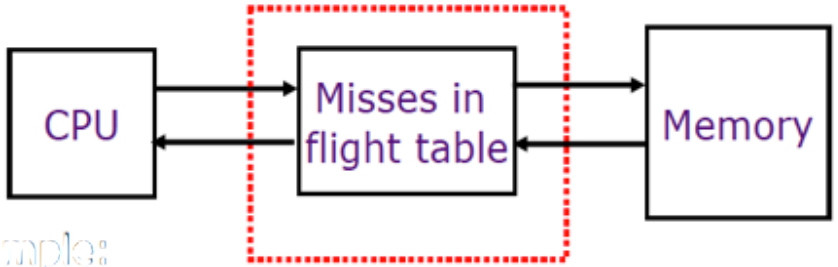
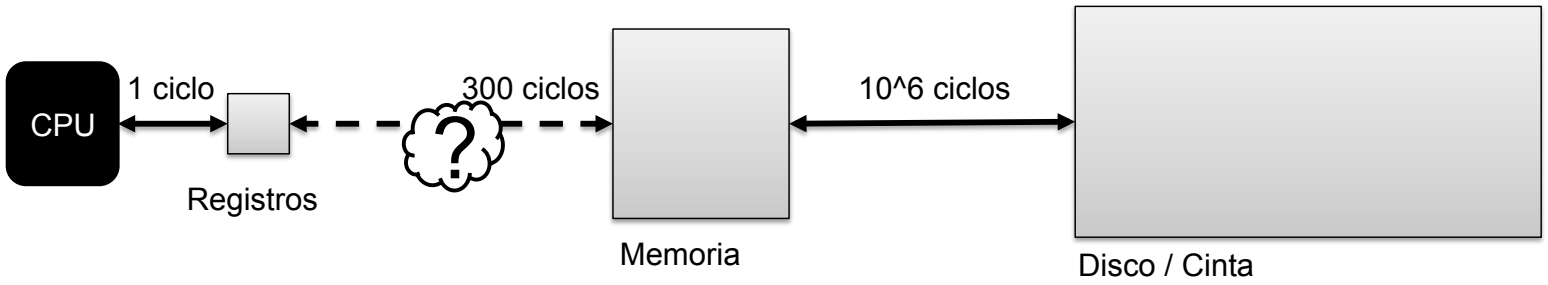


Imagen de Emer, 2005

- Ejemplo:
  - Suponiendo un ancho de banda infinito en la memoria.
  - 100 ciclos / referencia a memoria, 1 + 0.2 refs a memoria / instrucción.
  - ⇒ Tamaño de la Tabla = 1.2 * 100 = 120 entradas.
- 120 operaciones de memoria independientes en marcha!

JERARQUÍA DE MEMORIA

- Número de ciclos de CPU necesarios para alcanzar los diferentes niveles de memoria.
  - Falta algo que permita no tener que detener la CPU.



MEMORIA MULTINIVEL

- Estrategia:
  - Ocultar las latencias utilizando memorias pequeñas y rápidas denominadas caches.
- Caches:
  - Son un mecanismo para ocultar la latencia de acceso a memoria.
  - Basado en la observación experimental que indica que los patrones de acceso de las referencias a la memoria realizados por el procesador son a menudo altamente predecibles:

```
                    PC
      …             96
loop: ADD r2, r1, r1  100   Existe un patrón de acceso
      SUBI r3, r3, #1 104   de las instr a la memoria ?
      BNEZ r3, loop   108
      …               112
```

PATRONES TÍPICOS DE ACCESO A LA MEMORIA

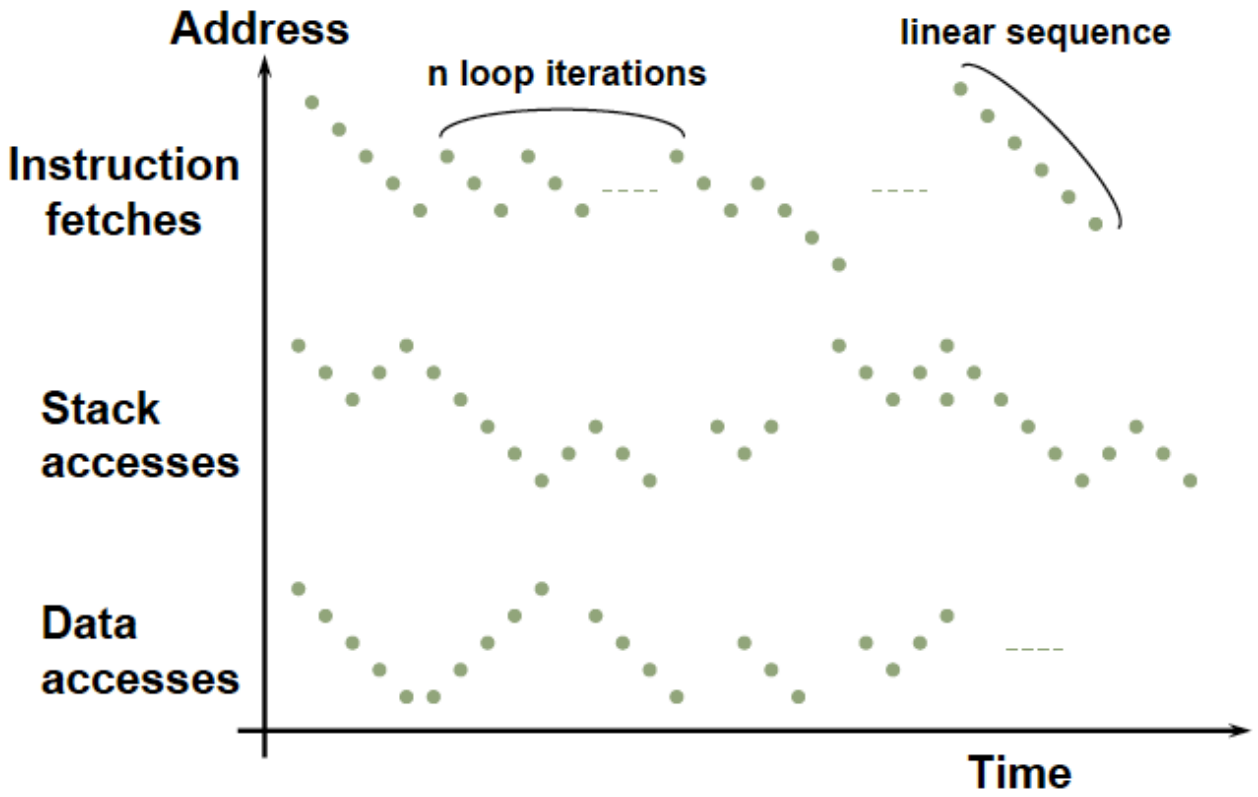


Imagen de Emer, 2005

FUNDAMENTO BÁSICO

- Principio de Localidad

  Los accesos a Memoria realizados por un programa no están uniformemente distribuidos.

  – Localidad Temporal: si una dirección es referenciada es muy probable que la misma dirección vuelva a ser referenciadas de nuevo muy pronto.

  – Localidad Espacial: si una dirección es referenciada es muy probable que las direcciones próximas a esta serán referenciadas de nuevo muy pronto.

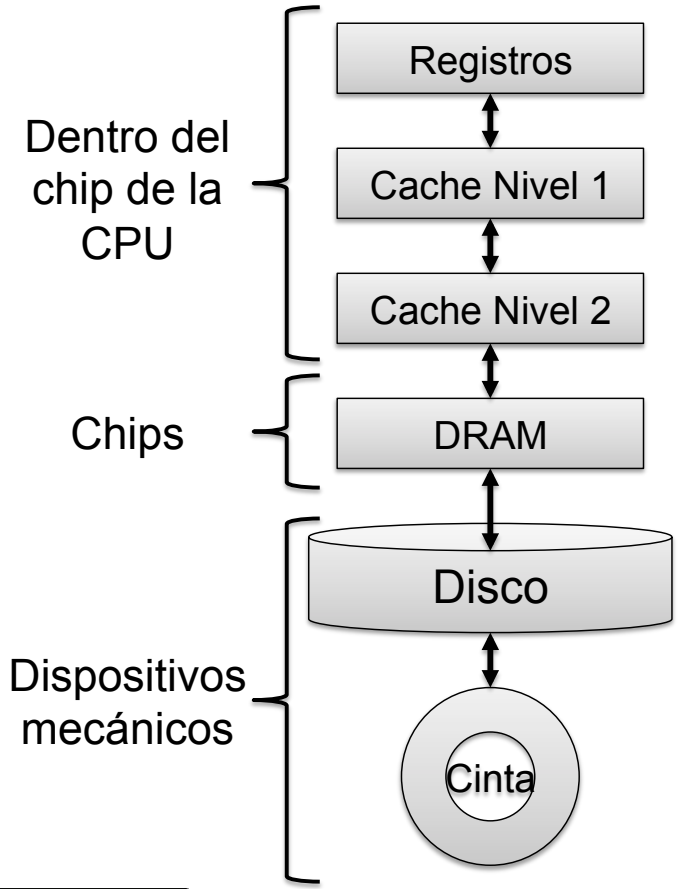
CACHÉS

- Las Cachés explotan ambos tipos de predecibilidad:
  - Explotan la localidad temporal recordando los contenidos de las posiciones de memoria accedidas recientemente.
  - Explotan la localidad espacial mediante búsquedas de bloques de datos alrededor de las posiciones de memoria recientemente accedidas.
- Suelen ser memorias pequeñas y rápidas (SRAM, static RAMs)
  - Tamaño: Registos << SRAM << DRAM
  - Latencia: Registos << SRAM << DRAM
  - Bandwidth: on-chip >> off-chip
- En un acceso a los datos:

  hit (data $\in$ fast memory) $\Rightarrow$ low latency access

  miss (data $\notin$ fast memory) $\Rightarrow$ long latency access (DRAM)
  - La cache es efectiva solo si el ancho de banda a la cache B es mucho mayor que el de acceso a la memoria principal A, B << A.

JERARQUÍA DE MEMORIA TÍPICA (ACTUAL)



	Tiempo de acceso	Capacidad	Gestionado por
Registros	1 ciclo	~500 Bytes	software/compilador
Cache Nivel 1	1-3 ciclos	~64 KB	hardware
Cache Nivel 2	5-10 ciclos	1 – 10 MB	hardware
DRAM	~100 ciclos	~10 GB	software/OS
Disco, Cinta	10^6 – 10^7 ciclos	~100 GB	software/OS

JERARQUÍA DE MEMORIA

- Los registros de la CPU sólo son visibles a nivel de leng. máquina.
- Las caches no son visibles.
- La memoria principal la ven programas y S.O.

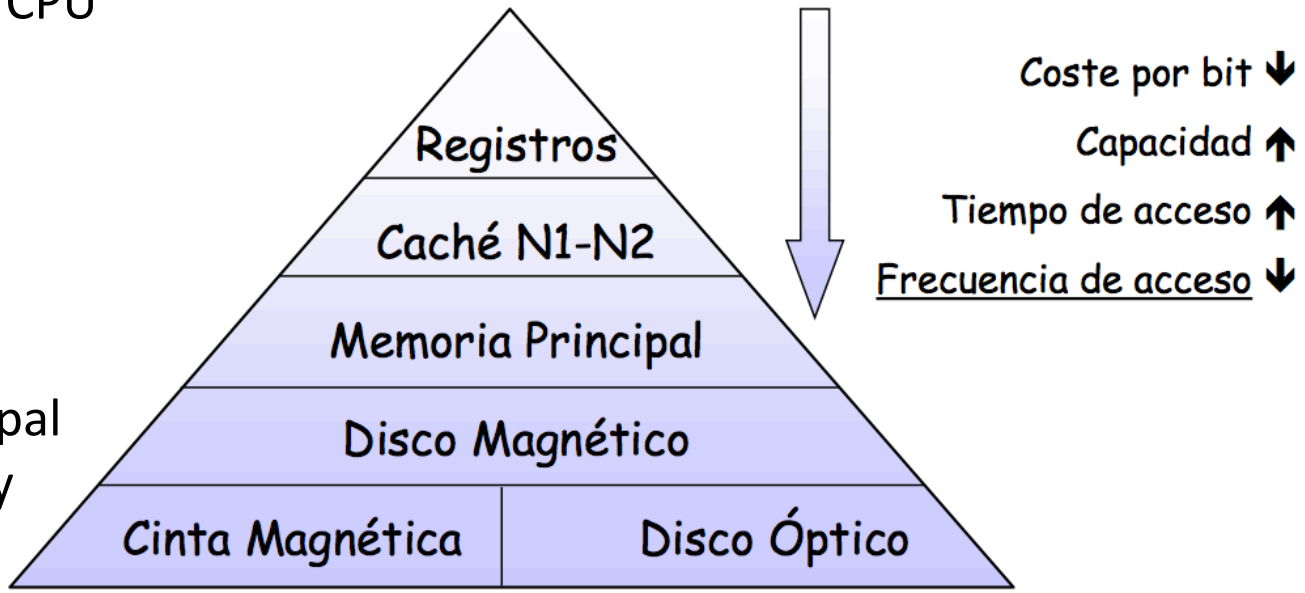


Imagen de Aylagas, 2013

JERARQUÍA DE MEMORIA TÍPICA

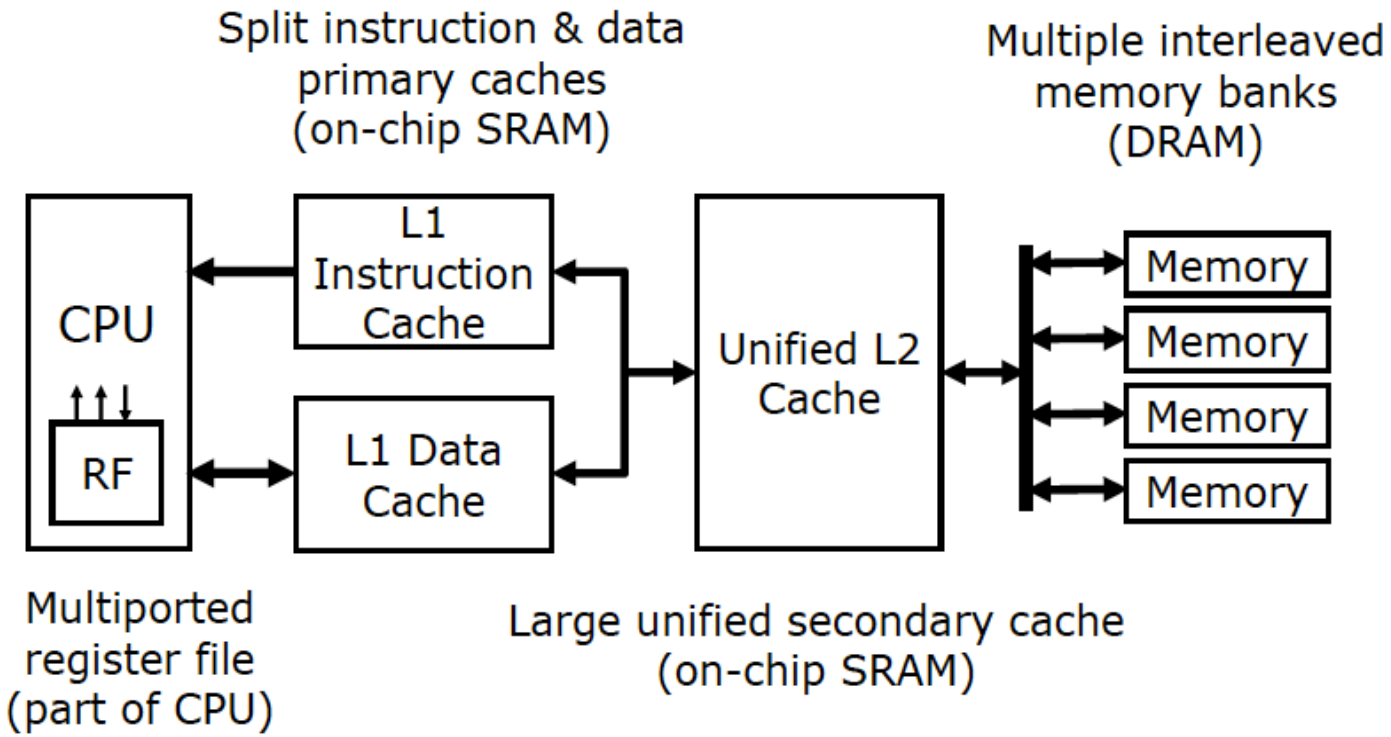


Imagen de Emer, 2005

TÉRMINOS USUALES

- Block – quantum of data within memory hierarchy.
- Page – contiguous block of virtual memory.
- Hit – A block is present at the searched level.
- Miss - A block is NOT present at the searched level.
- Hit time – the search time of a block (includes block access in case of hit).
- Miss penalty - time to replace a block from lower level, including time to provide to CPU.
- Miss time = hit time + miss penalty.
- Hit rate - fraction of accesses found in that level.
- Miss rate = 1- hit rate .
- Inclusion – every block in level j, also resides in levels j+1, j+2, …
- Exclusion – every block in level j, does NOT reside in levels j-1,j-2, …

Flujo de información

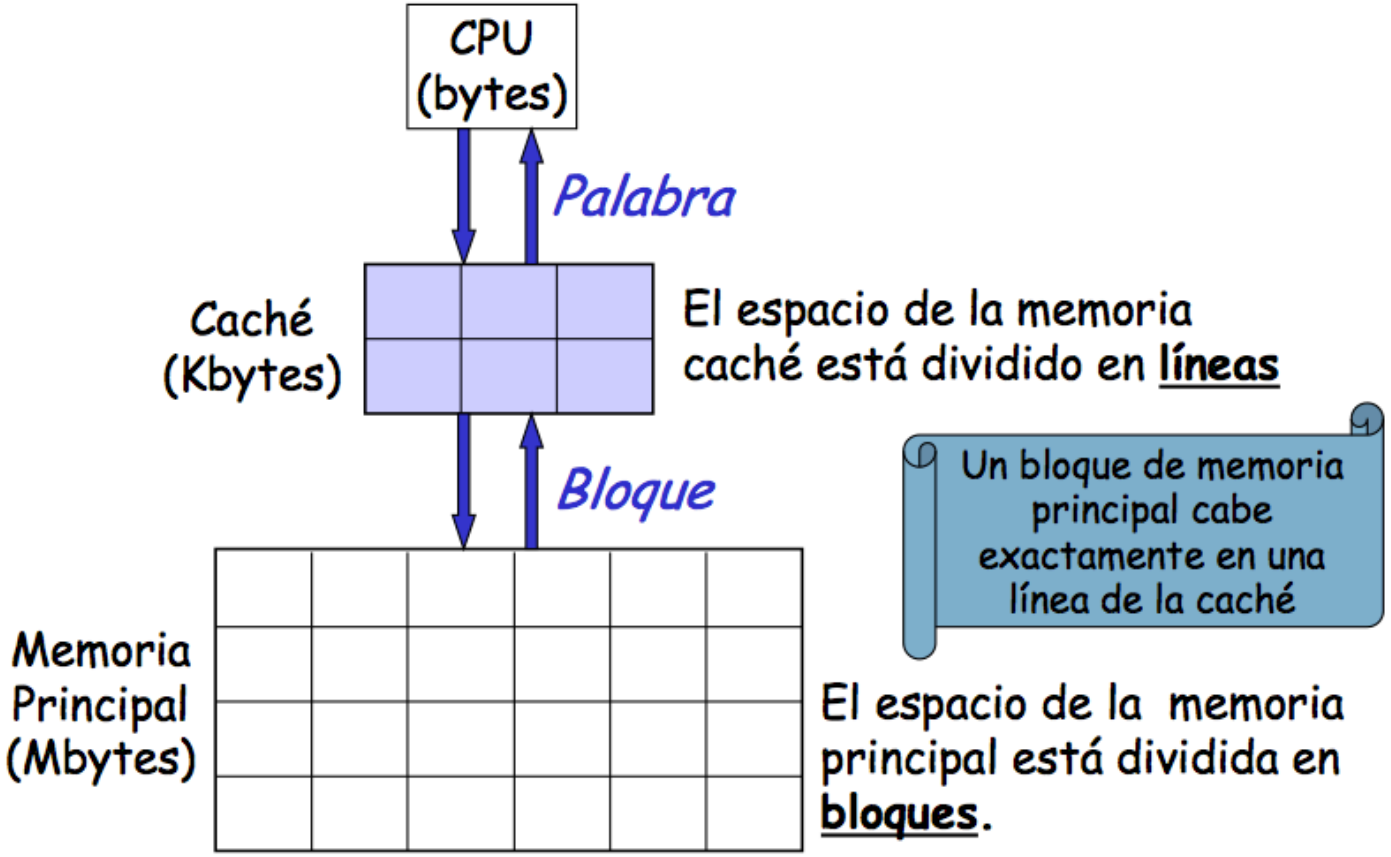
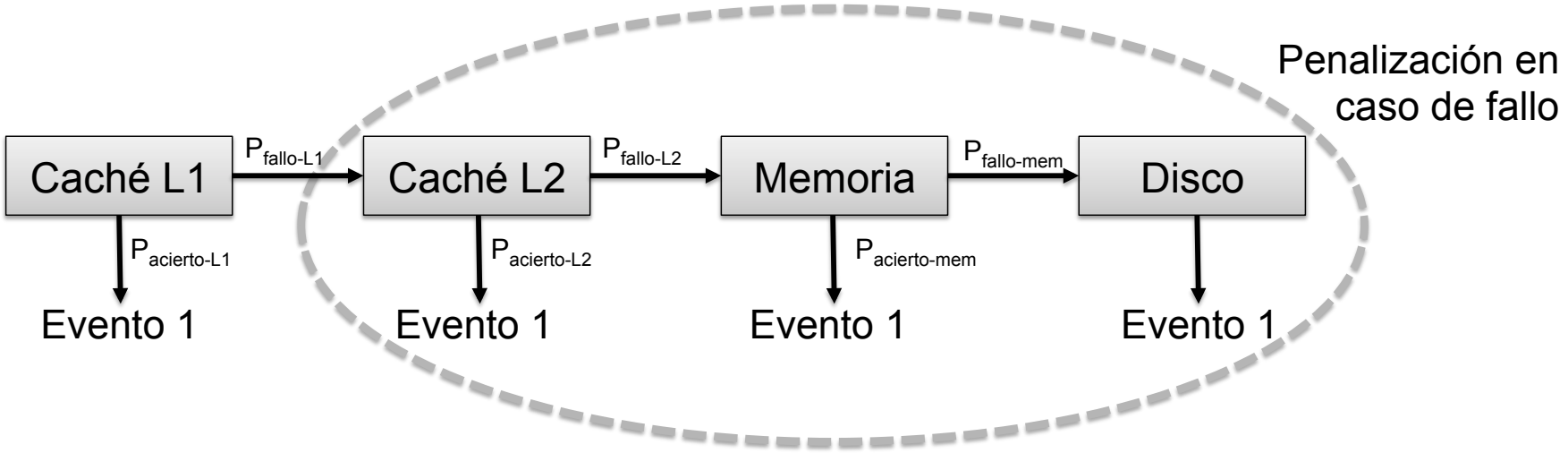


Imagen de Aylagas, 2013

GRAFO DE EVENTOS



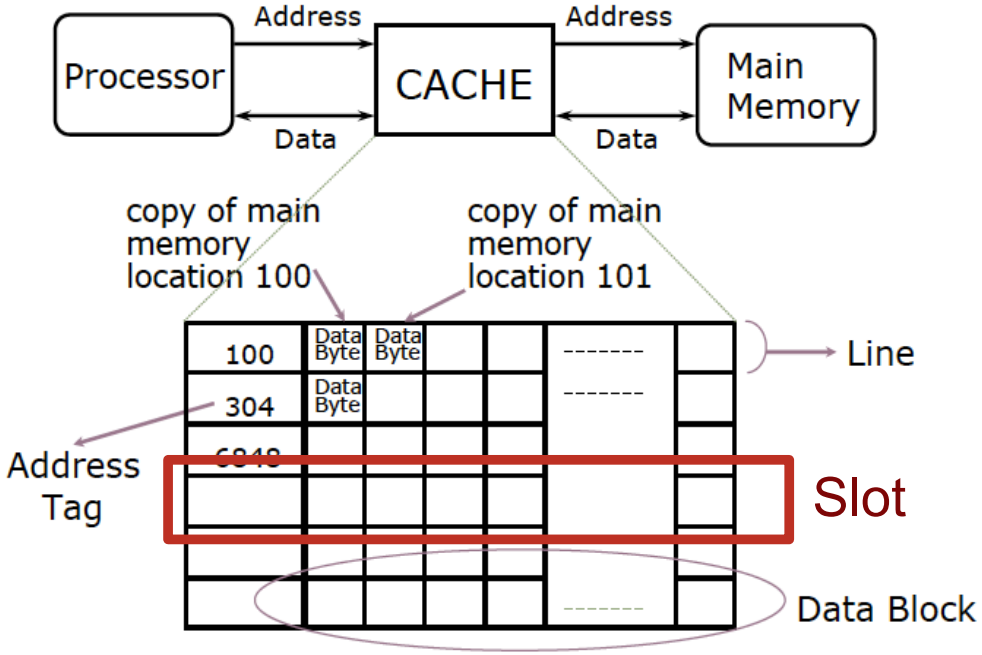
$P_{Evento1} = P_{acierto\text{-}L1}$

$P_{Evento2} = P_{fallo\text{-}L1} * P_{acierto\text{-}L2}$

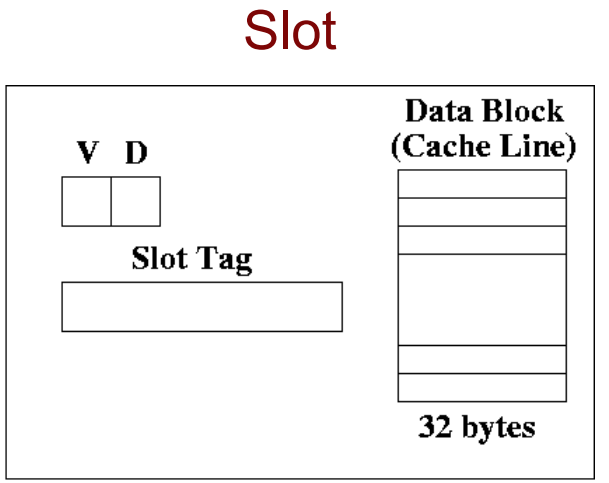
$P_{Evento3} = P_{fallo\text{-}L1} * P_{fallo\text{-}L2} * P_{acierto\text{-}mem}$

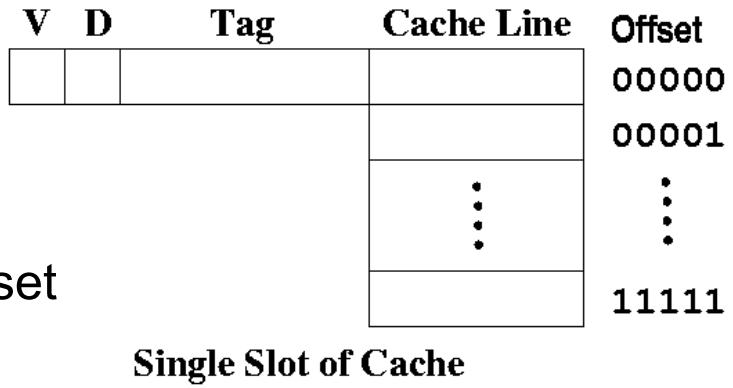
$P_{Evento4} = P_{fallo\text{-}L1} * P_{fallo\text{-}L2} * P_{fallo\text{-}mem}$

ESTRUCTURA DE LA CACHÉ



Imagenes de Emer, 2005





Single Slot of Cache

Main memory address => Tag, Slot position, offset

Data: block

POLÍTICA DE CORRESPONDENCIA O UBICACIÓN

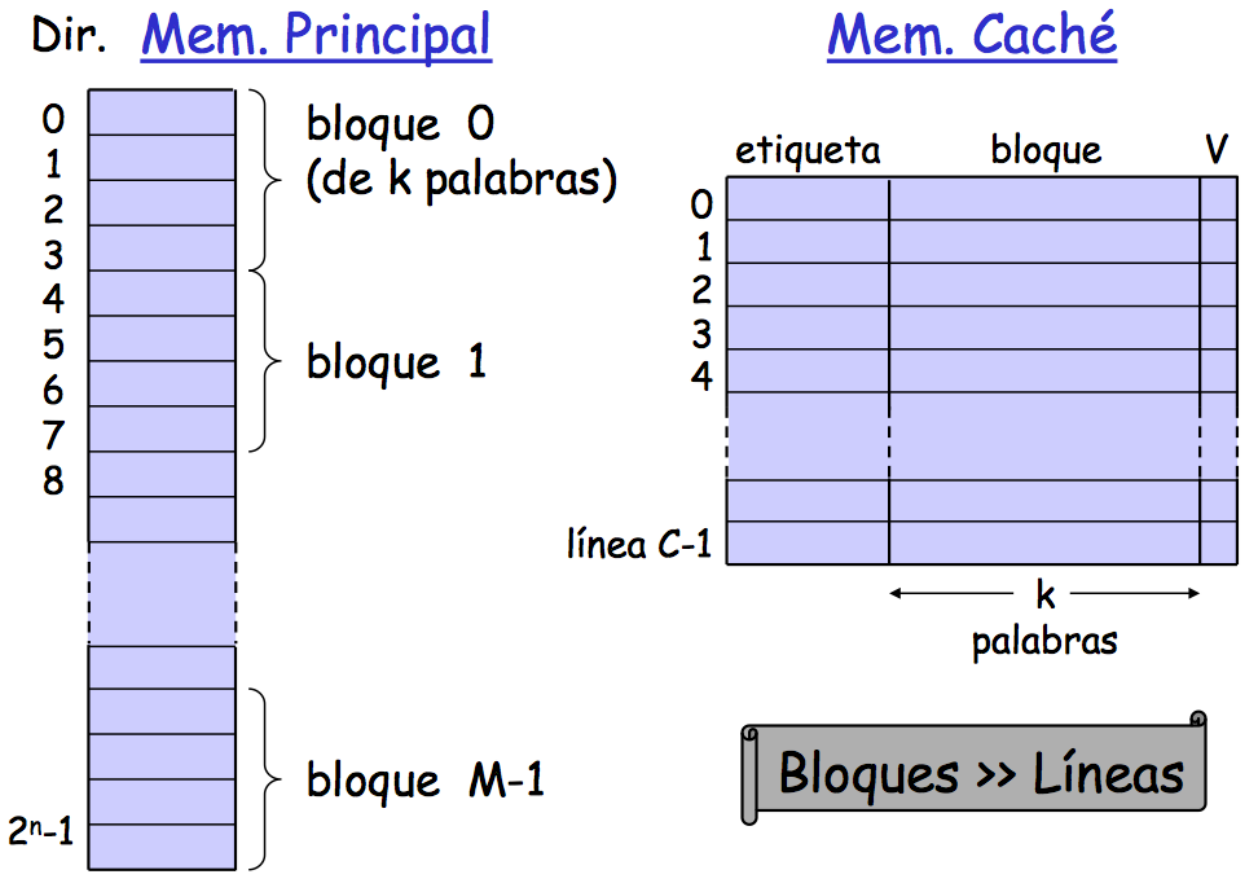
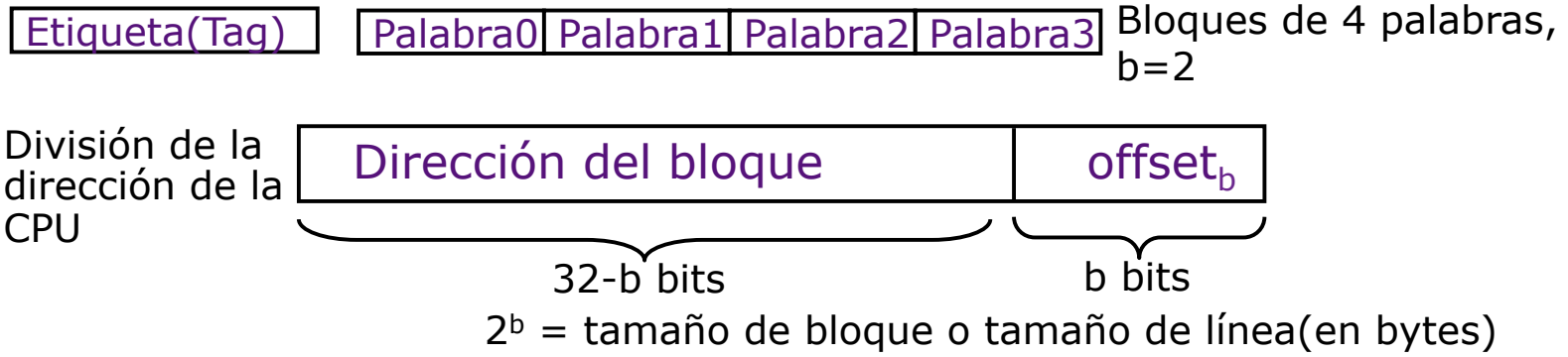


Imagen de Aylagas, 2013

TAMAÑO DE BLOQUE

- El Bloque es la unidad de transferencia de datos entre la cache y la memoria.



El número de bytes (Words) del bloque determina el número de bits dedicados al “offset”.

- Bloques grandes tienen diferentes ventajas hardware.
  - Menor overhead con la etiqueta (tag overhead).
  - Explota la rápida transferencia en ráfaga desde la DRAM.
  - Explota la rápida transferencia en ráfaga sobre buses anchos.

Lectura en una caché

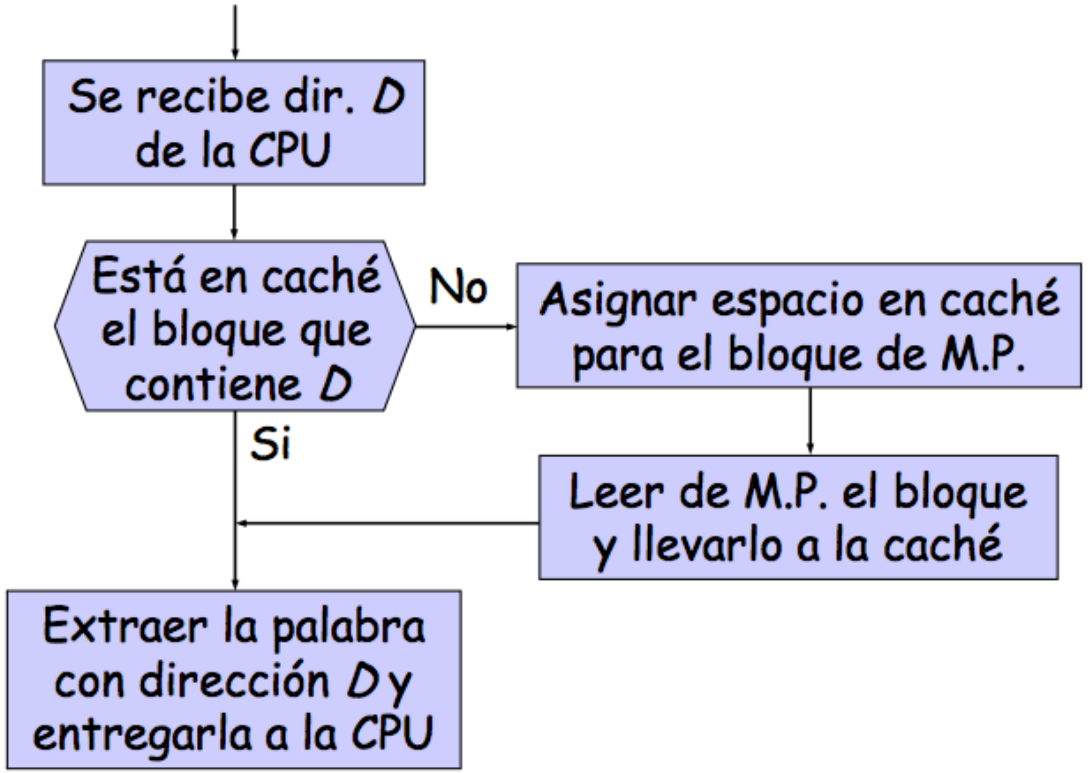


Imagen de Aylagas, 2013

Efectividad de la memoria Caché

Tiempo medio de acceso a memoria

EJEMPLO

$T_{MP} = 500$ ηs
$T_C = 50$ ηs
$P_A = 0{,}99$

Efectividad de la memoria Caché

Tiempo medio de acceso a memoria

$T_{acceso} = T_C \cdot P_A + T_{MP} \cdot (1 - P_a)$

P_A : Probabilidad de acierto

T_C : Tiempo de acceso a caché

T_{MP} : Tiempo de acceso a M. P.

EJEMPLO

$T_{MP} = 500$ ηs

$T_C = 50$ ηs

$P_A = 0{,}99$

Efectividad de la memoria Caché

Tiempo medio de acceso a memoria

$$T_{acceso} = T_C \cdot P_A + T_{MP} \cdot (1 - P_a)$$

P_A : Probabilidad de acierto

T_C : Tiempo de acceso a caché

T_{MP} : Tiempo de acceso a M. P.

EJEMPLO

$T_{MP} = 500$ ηs
$T_C = 50$ ηs
$P_A = 0{,}99$

$$T_{acceso} = 50 \cdot 0{,}99 + 500 \cdot 0{,}01 = 54{,}5 \text{ ηs}$$

¿ Merece la pena la caché ?

$$\text{Índice de mejora} = \frac{T_{sin\ caché}}{T_{con\ caché}} = \frac{500}{54{,}4} = 9{,}19$$

PARÁMETROS DE DISEÑO DE LA CACHÉ

- Tamaño de la cache y del bloque.
- Número de Cachés.
- Función de correspondencia o ubicación.
- Algoritmo de reemplazo de bloques.
- Política de escritura.
- Tamaño de línea.

TAMAÑO DE LA CACHÉ

- En general cuanto mayor es la caché más puertas se requieren para direccionarla por lo que tienden a ser algo más lentas (incluso a igualdad de tecnología).
- Con frecuencia su tamaño está limitado por el espacio disponible en el chip, ya que las caches de primer nivel están integradas.
- El tamaño óptimo suele depender del tipo de tarea que realiza el programa por lo que es difícil de optimizar.
- Tamaños típicos:
  - Nivel L1: 8-64 KB
  - Nivel L2: 256 KB – 4 MB
  - Nivel L3 (menos usual): 2 MB – 36 MB

Aparición de las cachés

	INFORMACIÓN DE MEMORIA CACHÉ Y VELOCIDAD DE ALGUNOS PROCESADORES INTEL			
	MODELO	VELOCIDAD EN MHz	CACHÉ L1 EN KB	CACHÉ L2 EN KB
1979	8088	8	0	0
	8086	8	0	0
	80386DX	40	0	0
1989	80486SLC	25	8	0
	Pentium	75	16	0
		100	16	0
1997	Pentium II	233	32	512
		450	32	512
	Pentium III	450	32	512
		1000	32	256
	Pentium 4	1400	32	256
		2000	32	512

NÚMERO DE CACHÉS

- En origen sólo se introducía una caché , pero en la actualidad se usan múltiples cachés, lo que requiere un diseño equilibrado del conjunto de las mismas.
- Niveles de Caché:
  - Las cachés L1 (on chip) suelen estar integradas en el mismo chip que el procesador.
    - Reduce mucho el tráfico de datos al bus externo del procesador lo que incrementa las prestaciones globales del sistema.
  - Las cachés L2 , suelen ser externas al chip del procesador (en los multicores también están en el chip).
    - Mejora la rapidez de acceso a la memoria principal (DRAM) sobre todo si se usa una SRAM para la caché. Con frecuencia el acceso a esta caché no usa los buses del sistema para reducir el tráfico en los mismos.
    - La tendencia es a incorporar la L2 también en el chip del procesador.
  - Las cachés L3, aparecen externamente al procesador cuando la de L2 se introduce en el chip, aunque algunos la integran ya en el chip.

CACHÉS UNIFICADAS VS SEPARADAS

- Las primeras cachés tenían un diseño unificado tanto para referencias a datos como a instrucciones.
- Sin embargo hoy en día las cachés de nivel L1 situadas en el chip suelen estar separadas, de forma que una contiene referencias a instrucciones y otra a los datos en uso.
- Las cachés L2 y L3 suelen tener diseño unificado, estén o no situadas dentro del chip.

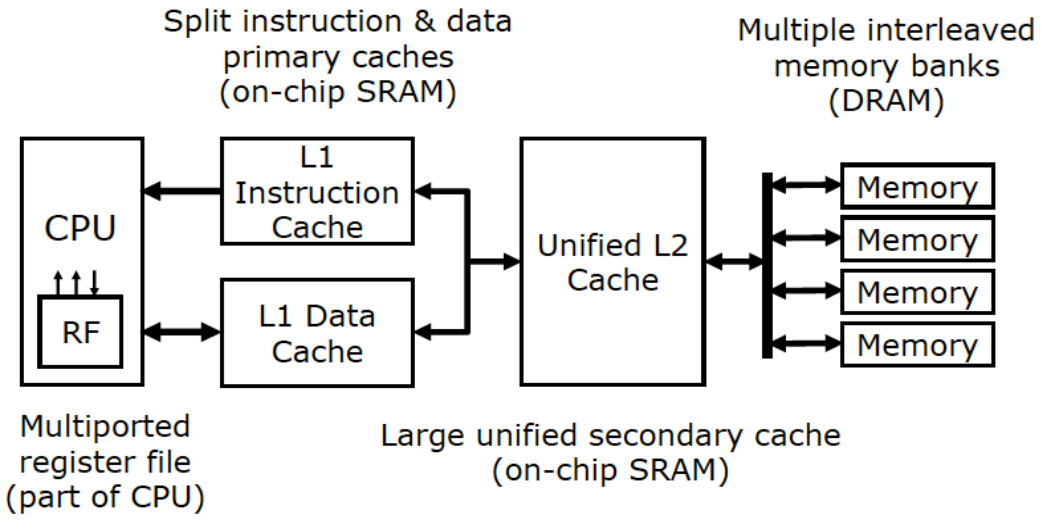
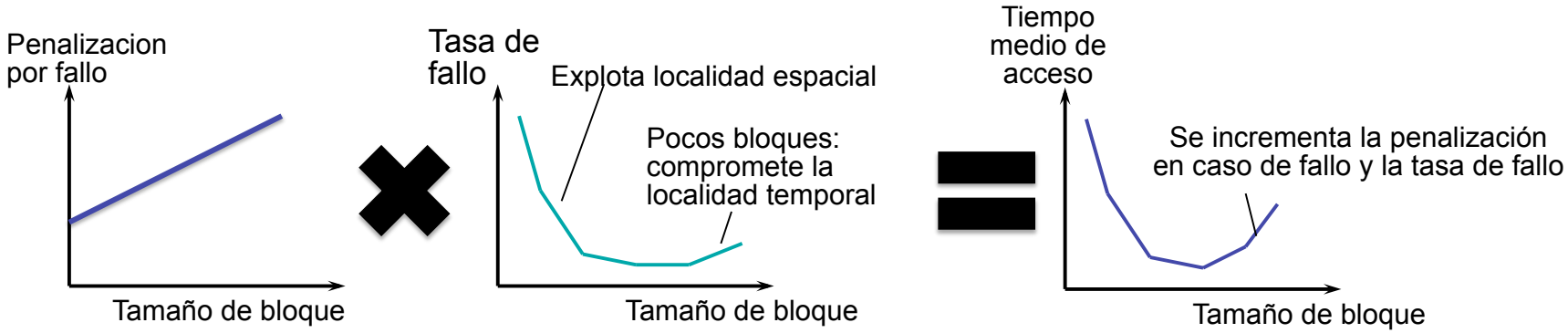


Imagen de Emer, 2005

TAMAÑO DE BLOQUE VS CACHÉ

- El incremento del tamaño del bloque incrementa por lo general la penalización por fallo y disminuye la tasa de fallos.
- Eventualmente: disminuye el retorno; menos bloques independientes; mayor tiempo de transferencia.



TAMAÑO DE BLOQUE VS CACHÉ

- **Benefits of Larger Block Size**
  - Spatial Locality: if we access a given word, we're likely to access other nearby words soon
  - Very applicable with Stored-Program Concept: if we execute a given instruction, it's likely that we'll execute the next few as well
  - Works nicely in sequential array accesses too

- **Drawbacks of Larger Block Size**
  - If block size is too big relative to cache size, then there are too few blocks
    - Result: miss rate goes up
  - Larger block size means larger miss penalty
    - on a miss, takes longer time to load a new block from next level (more data to load)

TAMAÑO DE BLOQUE VS CACHÉ

Tamaño bloque	Tamaño de la Caché				
	1K	4K	16K	64K	256K
16	15,05%	8,57%	3,94%	2,04%	1,09%
32	**13,34%**	7,24%	2,87%	1,35%	0,70%
64	13,76%	**7,00%**	**2,64%**	1,06%	0,51%
128	16,64%	7,78%	2,77%	**1,02%**	**0,49%**
256	22,01%	9,51%	3,29%	1,15%	0,49%

Tasa de faltas de caché, según el tamaño de bloque, para distintos tamaños de caché

Tamaño bloque	Tiempo por falta	Tamaño de la Caché				
		1K	4K	16K	64K	256K
16	42	7,321	4,599	2,655	1,859	1,458
32	44	**6,870**	**4,186**	**2,263**	1,594	1,308
64	48	7,605	4,360	2,267	**1,509**	**1,245**
128	56	10,318	5,357	2,551	1,571	1,274
256	72	16,847	7,847	3,369	1,828	1,353

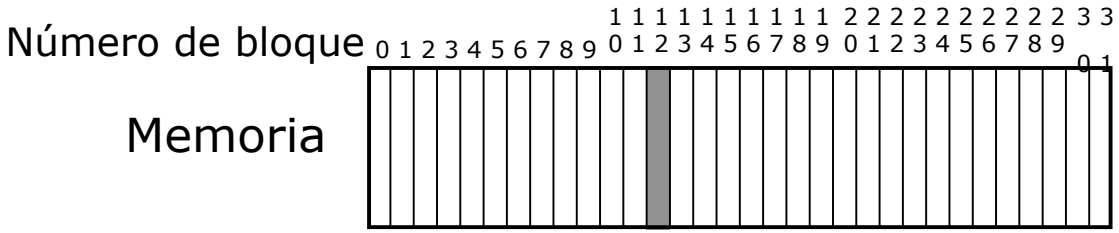
Tiempo medio de acceso a memoria, según el tamaño de bloque, para distintos tamaños de caché

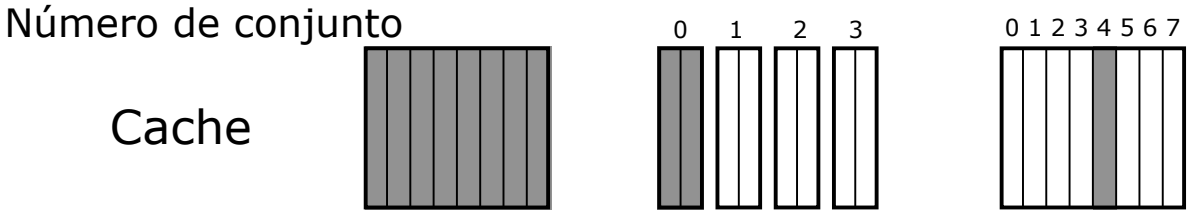
Imagenes de Aylagas, 2013

POLÍTICA DE CORRESPONDENCIA O UBICACIÓN

- Dado que las cachés son más pequeñas que la memoria principal es necesario algún tipo de algoritmo que permita establecer la correspondencia entre los bloques de la memoria principal y los de la caché (líneas).
- Las políticas de correspondencia más usuales son:
  - Mapeo directo, hace corresponder cada bloque de la memoria principal a una sola línea posible de la cache (i=j mod m).
  - Mapeo asociativo, aquí se permite que cada bloque de memoria pueda cargarse en una posición cualquiera de la caché.
  - Mapeo asociativo por bloques, es una solución intermedia entre las dos anteriores.

POLÍTICA DE UBICACIÓN DE BLOQUES DE DATOS

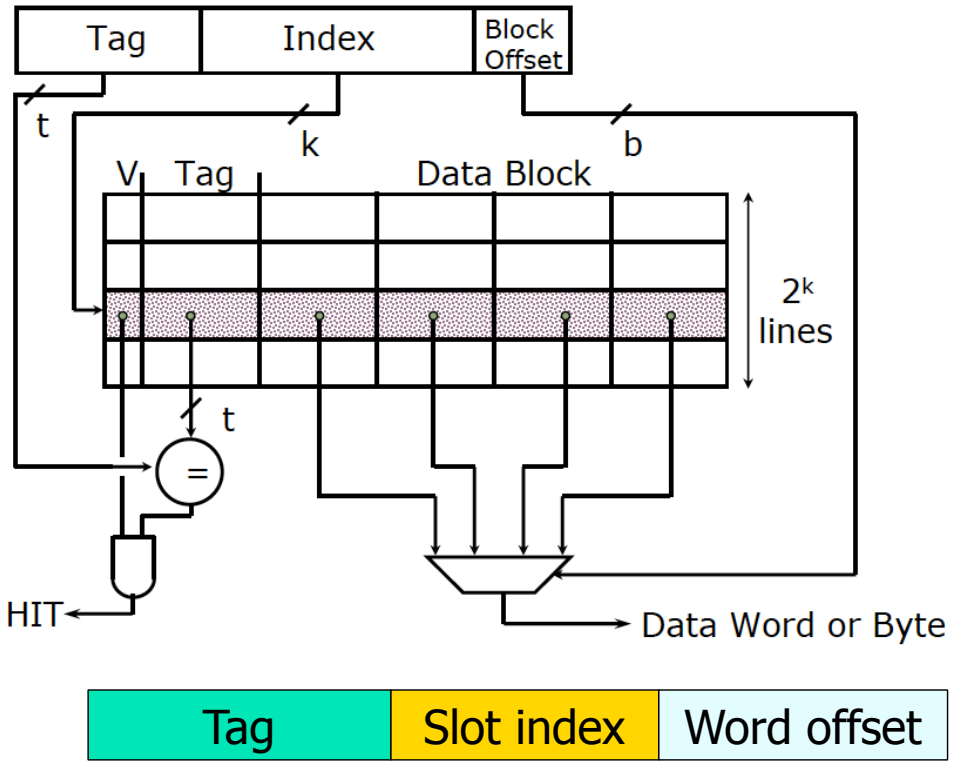




	Totalmente asociativa	Asociativa por conjuntos (2-vías)	Mapeo directo
El bloque 12 se ubicaría:	En cualquier parte	En cualquier parte del conjunto 0 *(12 mod 4)*	Solo en el bloque 4 *(12 mod 8)*

MAPEO DIRECTO (DIRECT MAPPED CACHE)

Dirección de memoria

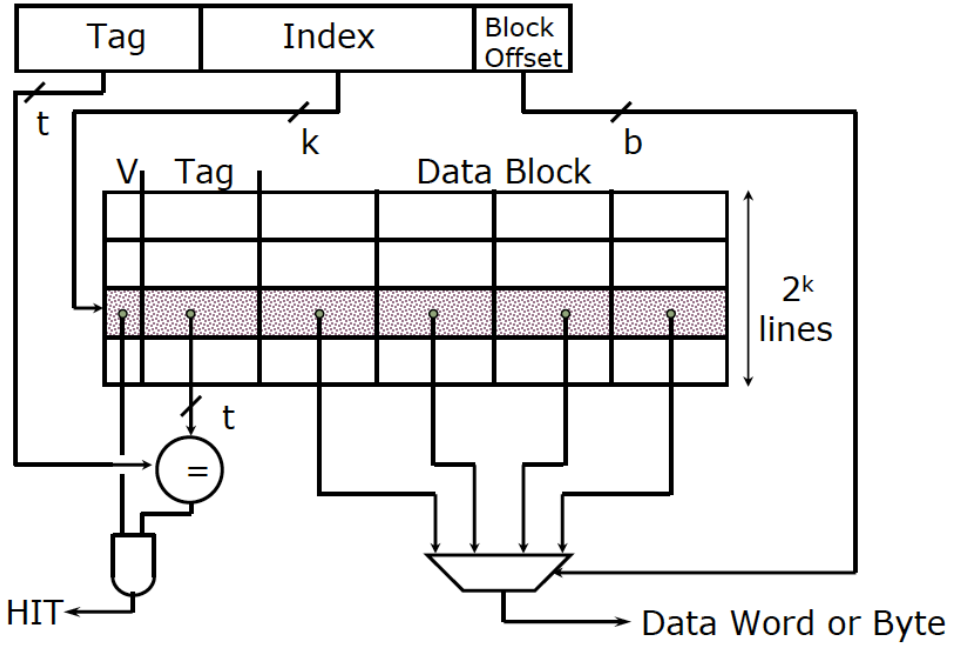


- Memory: M blocks (4K)
- Block size: B words (16)
- Cache: N blocks (128)

Imagen de Emer, 2005

MAPEO DIRECTO (DIRECT MAPPED CACHE)

Dirección de memoria

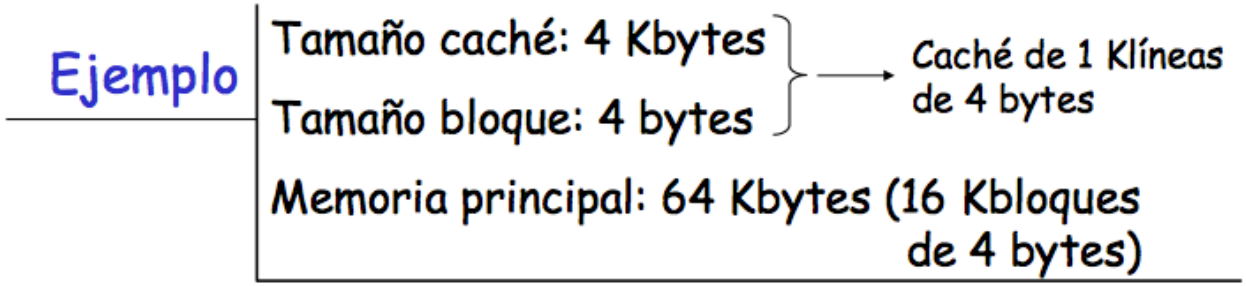


- Memory: M blocks (4K)
- Block size: B words (16)
- Cache: N blocks (128)
- Address size: $\log_2 (M * B)$

$= \log_2 (4k * 16) = 16$

Tag	Slot index	Word offset
Remaining bits $\log_2 M/N$	$\log_2 N$	$\log_2 B$
5	7	4

MAPEO DIRECTO (DIRECT MAPPED CACHE) – EJEMPLO



Ejemplo de Aylagas, 2013

MAPEO DIRECTO (DIRECT MAPPED CACHE) – EJEMPLO

Ejemplo

Tamaño caché: 4 Kbytes
Tamaño bloque: 4 bytes
→ Caché de 1 Klíneas de 4 bytes

Memoria principal: 64 Kbytes (16 Kbloques de 4 bytes)

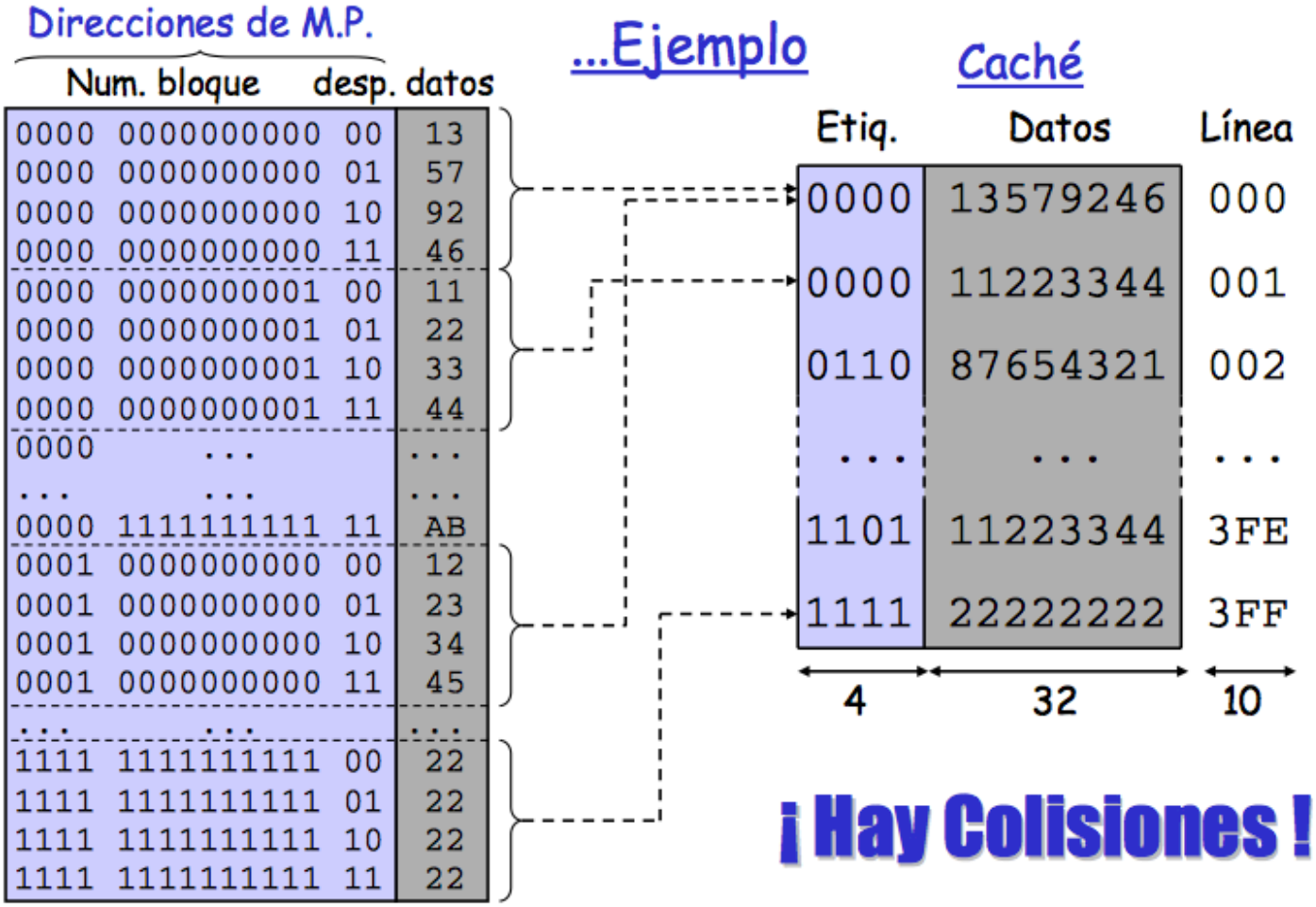
Dirección M.P.

Número de bloque		despl.
4	10	2
etiqueta	línea	

Línea caché	Bloques de memoria principal
0	0, 400, 800, C00, ..., 3C00
1	1, 401, 801, C01, ..., 3C01
...	...
3FF	3FF, 7FF, BFF, ..., 3FFF

Ejemplo de Aylagas, 2013

MAPEO DIRECTO (DIRECT MAPPED CACHE) – EJEMPLO



Ejemplo de Aylagas, 2013

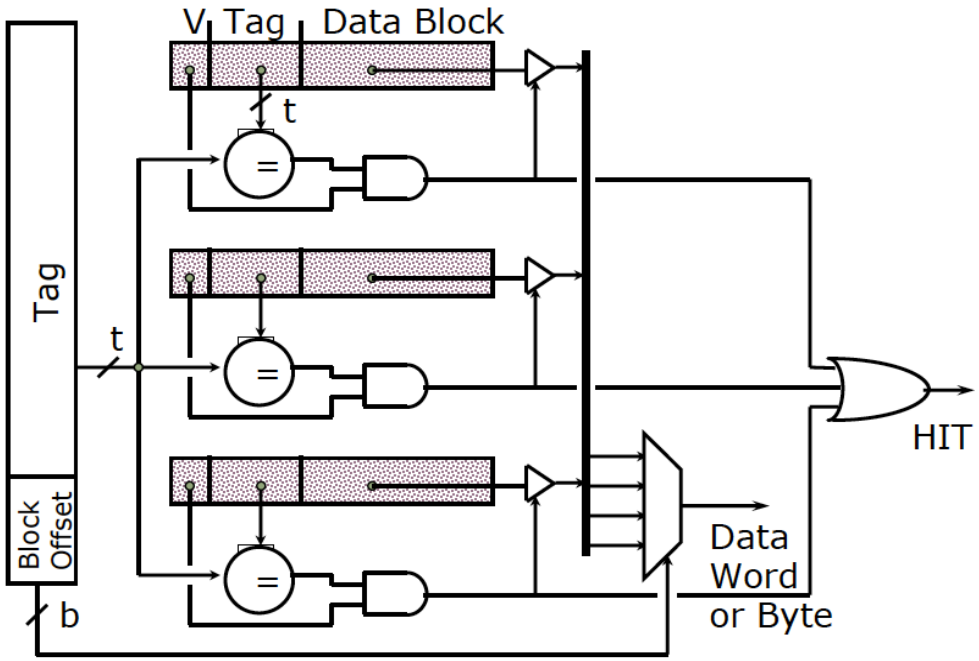
POLÍTICA DE MAPEO DIRECTO: VENTAJAS Y DESVENTAJAS

- Ventajas:
  - Es simple y poco costosa de implantar.

- Desventajas:
  - Hay una posición concreta en la cache para cada bloque de la memoria.
  - Si un programa referencia repetidamente a dos bloques de memoria asignados a la misma caché se produce un intercambio contínuo y cae la tasa de aciertos, este fenómeno se conoce como “thrashing”.

MAPEO TOTALMENTE ASOCIATIVO

Dirección de memoria



Aquí la dirección de memoria se interpreta como una etiqueta y un offset a la palabra concreta dentro del bloque.

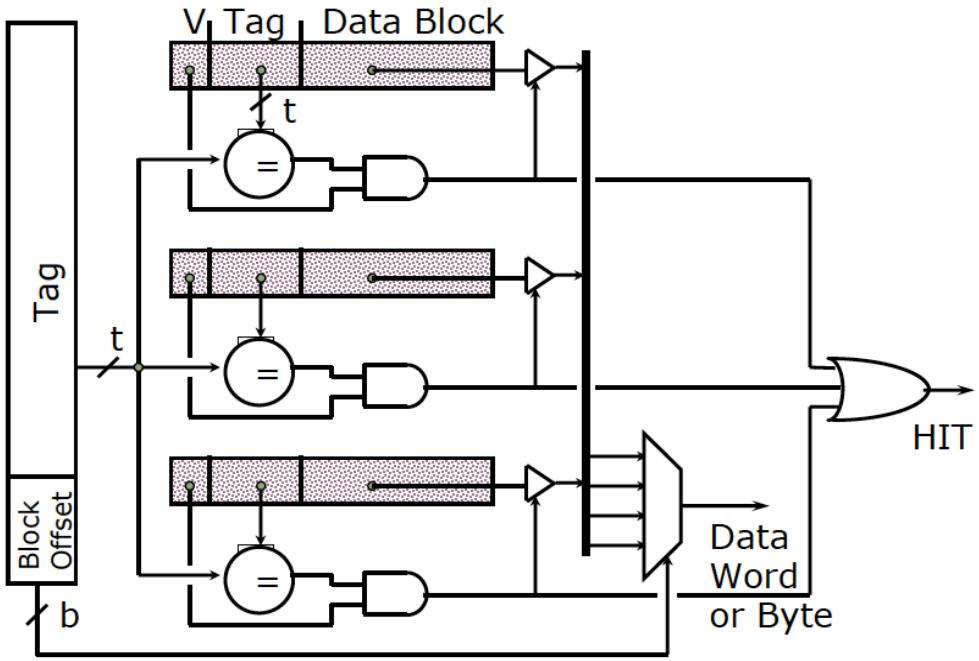
El número de líneas de la caché puede fijarse arbitrariamente.

- Memory: M blocks (4K)
- Block size: B words (16)
- Cache: N blocks (128)

Imagen de Emer, 2005

MAPEO TOTALMENTE ASOCIATIVO

Dirección de memoria



Aquí la dirección de memoria se interpreta como una etiqueta y un offset a la palabra concreta dentro del bloque.

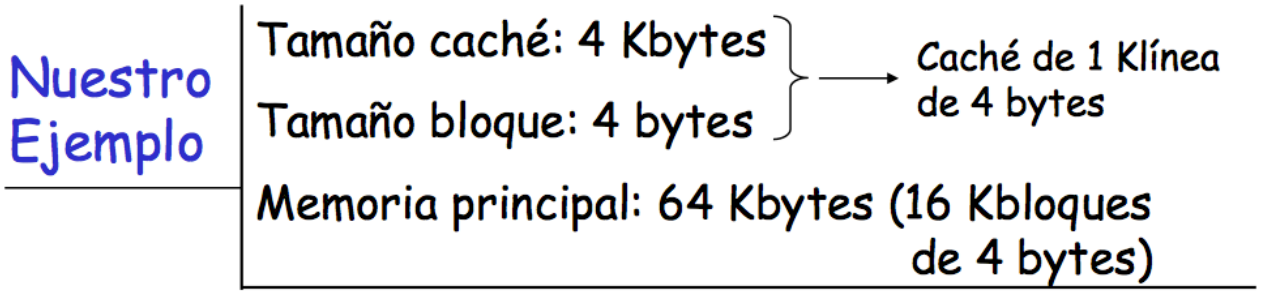
El número de líneas de la caché puede fijarse arbitrariamente.

- Memory: M blocks (4K)
- Block size: B words (16)
- Cache: N blocks (128)
- Address size: $\log_2$ (M * B)

= $\log_2$ (4k * 16) = 16

Tag	Word
Remaining bits $\log_2$ M	$\log_2$ B
12	4

MAPEO TOTALMENTE ASOCIATIVO - Ejemplo



Ejemplo de Aylagas, 2013

MAPEO TOTALMENTE ASOCIATIVO - Ejemplo

Nuestro Ejemplo

Tamaño caché: 4 Kbytes
Tamaño bloque: 4 bytes
→ Caché de 1 Klínea de 4 bytes

Memoria principal: 64 Kbytes (16 Kbloques de 4 bytes)

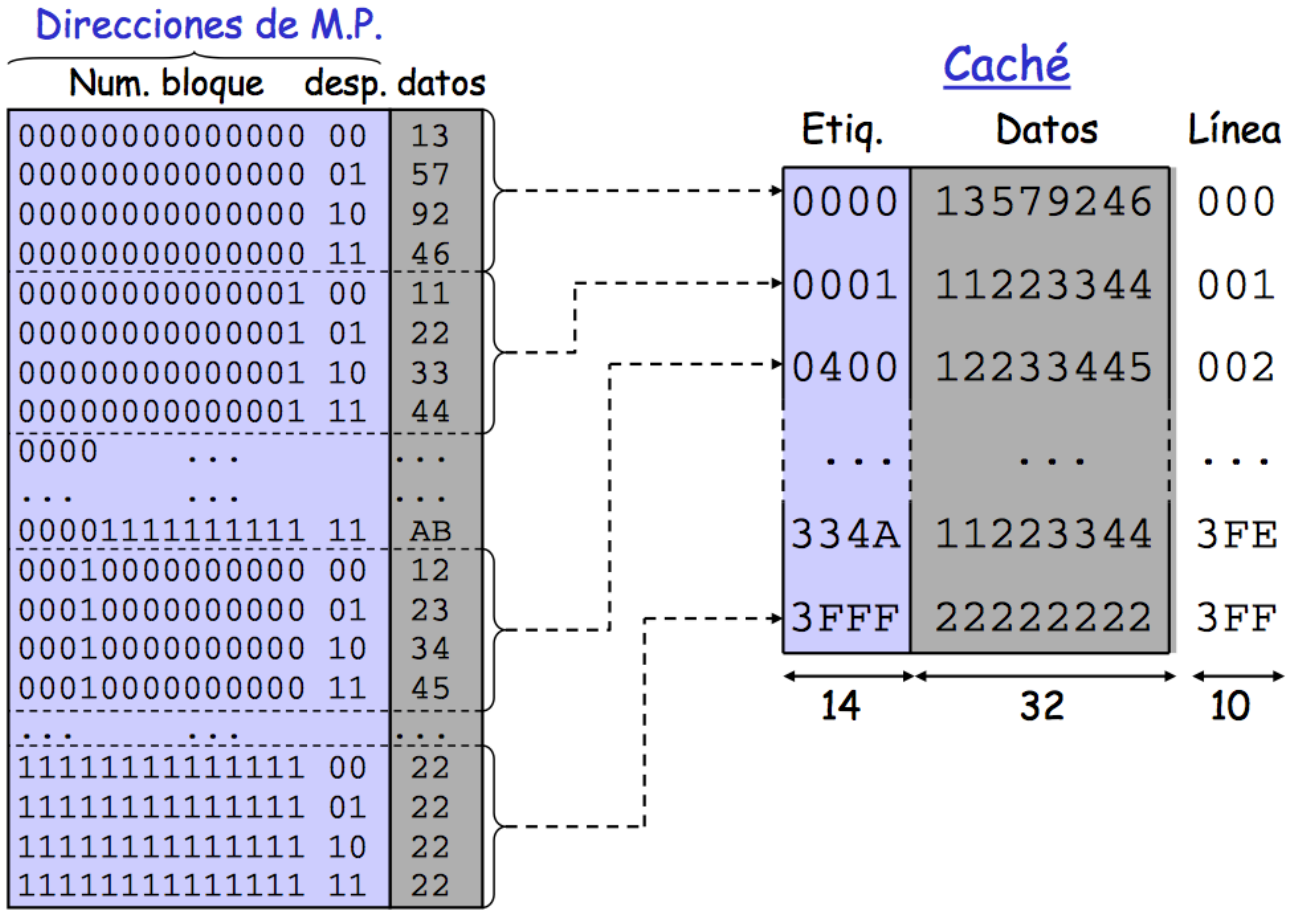
Dirección M.P.

Número de bloque (etiqueta)	despl.
14	2

Línea caché	Bloques de memoria principal
0	cualquiera
1	cualquiera
...	...
3FF	cualquiera

Ejemplo de Aylagas, 2013

MAPEO TOTALMENTE ASOCIATIVO - Ejemplo



Ejemplo de Aylagas, 2013

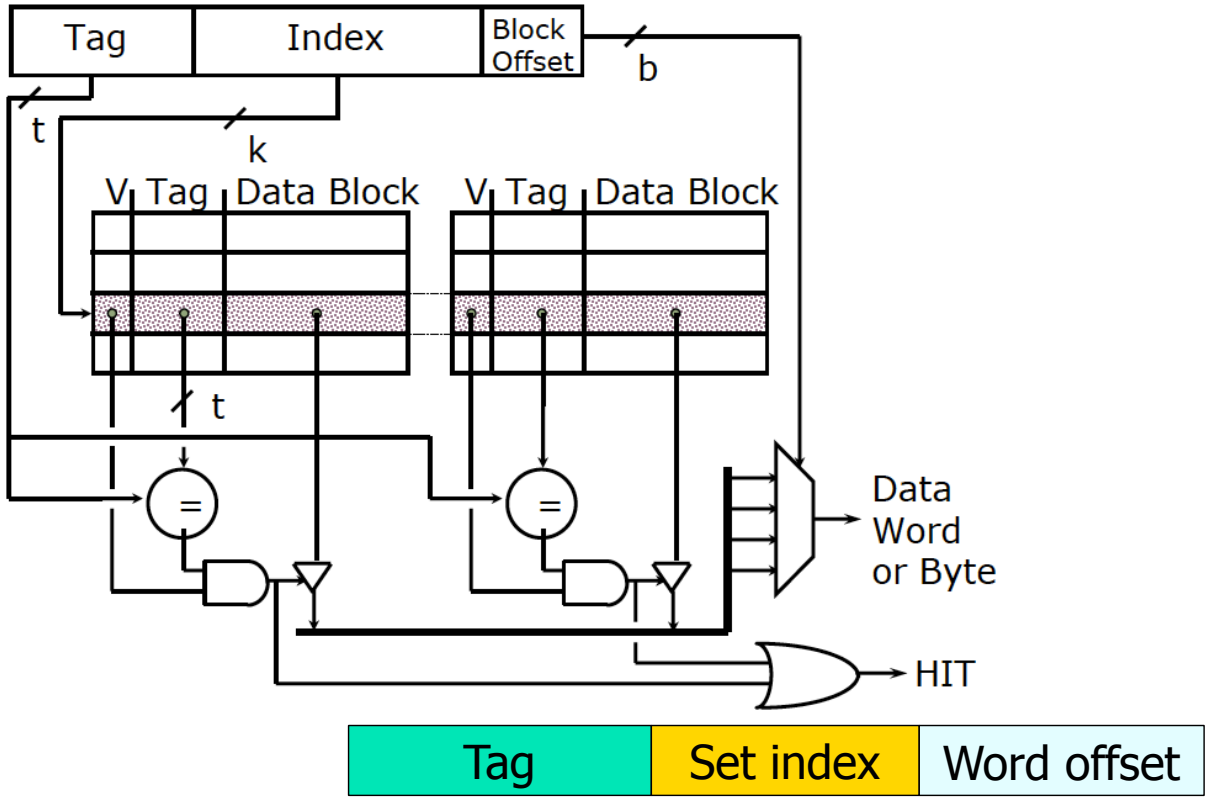
POLÍTICA DE MAPEO ASOCIATIVO: VENTAJAS Y DESVENTAJAS

- Ventajas:
  - La ubicación en la cache para cada bloque de la memoria es arbitraria, puede escribirse en cualquier línea libre por lo que disminuyen los fallos.

- Desventajas:
  - La circuitería es mucho más compleja que en el caso de mapeo directo.

CACHÉ ASOCIATIVA POR CONJUNTOS DE 2 VÍAS

Dirección de memoria

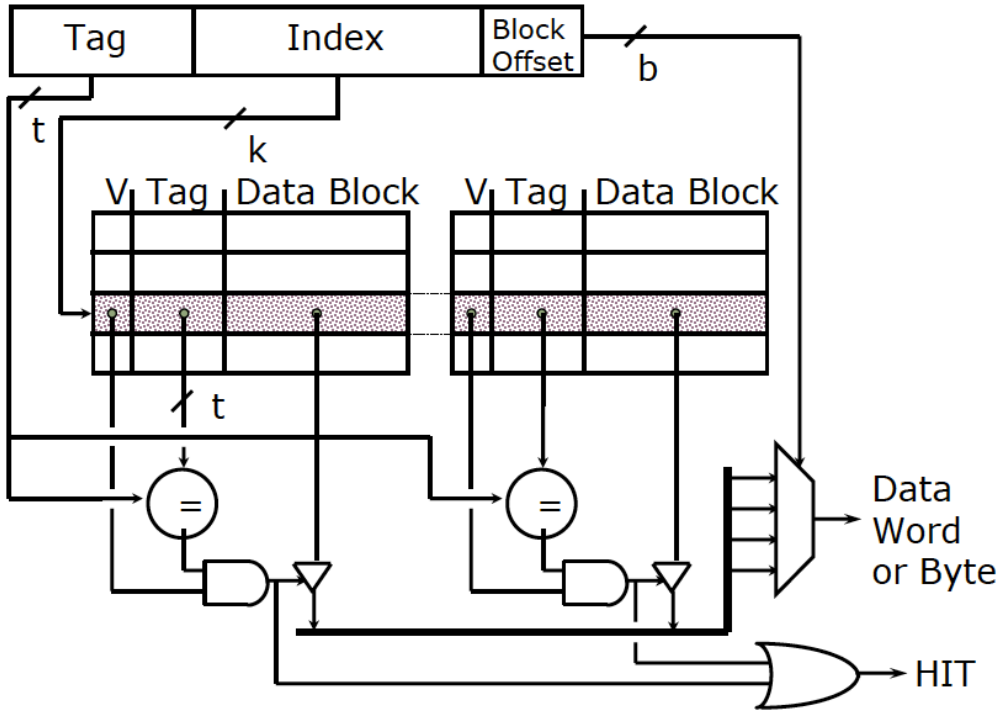


- Memory: M blocks (4k)
- Block size: B words (16)
- Cache: N blocks (128)
- Number of sets: S sets (32)
- Num of blocks per set: N (4)

Imagen de Emer, 2005

CACHÉ ASOCIATIVA POR CONJUNTOS DE 2 VÍAS

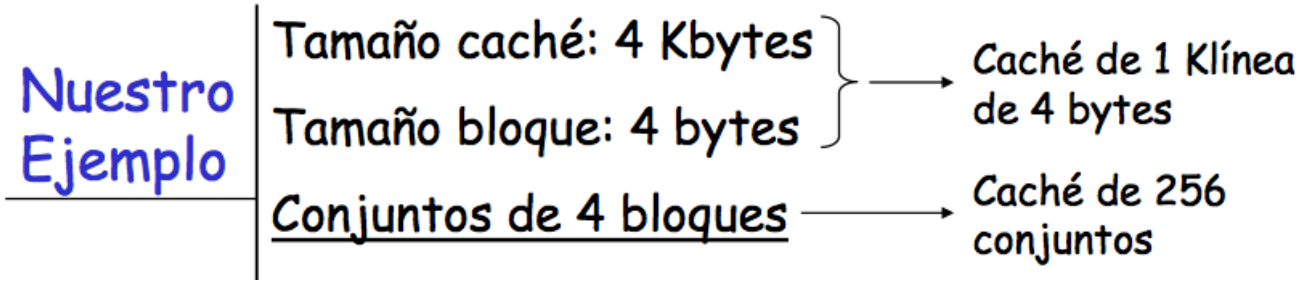
Dirección de memoria



- Memory: M blocks (4k)
- Block size: B words (16)
- Cache: N blocks (128)
- Number of sets: S sets (32)
- Num of blocks per set: N (4)
- Address size: $\log_2$ (M * B) (16)

Tag	Set index	Word offset
Remaining bits $\log_2$ M/S	$\log_2$ S	$\log_2$ B
7	5	4

CACHÉ ASOCIATIVA POR CONJUNTOS DE 2 VÍAS - Ejemplo



CACHÉ ASOCIATIVA POR CONJUNTOS DE 2 VÍAS - Ejemplo

Nuestro Ejemplo

- Tamaño caché: 4 Kbytes
- Tamaño bloque: 4 bytes

→ Caché de 1 Klínea de 4 bytes

- Conjuntos de 4 bloques

→ Caché de 256 conjuntos

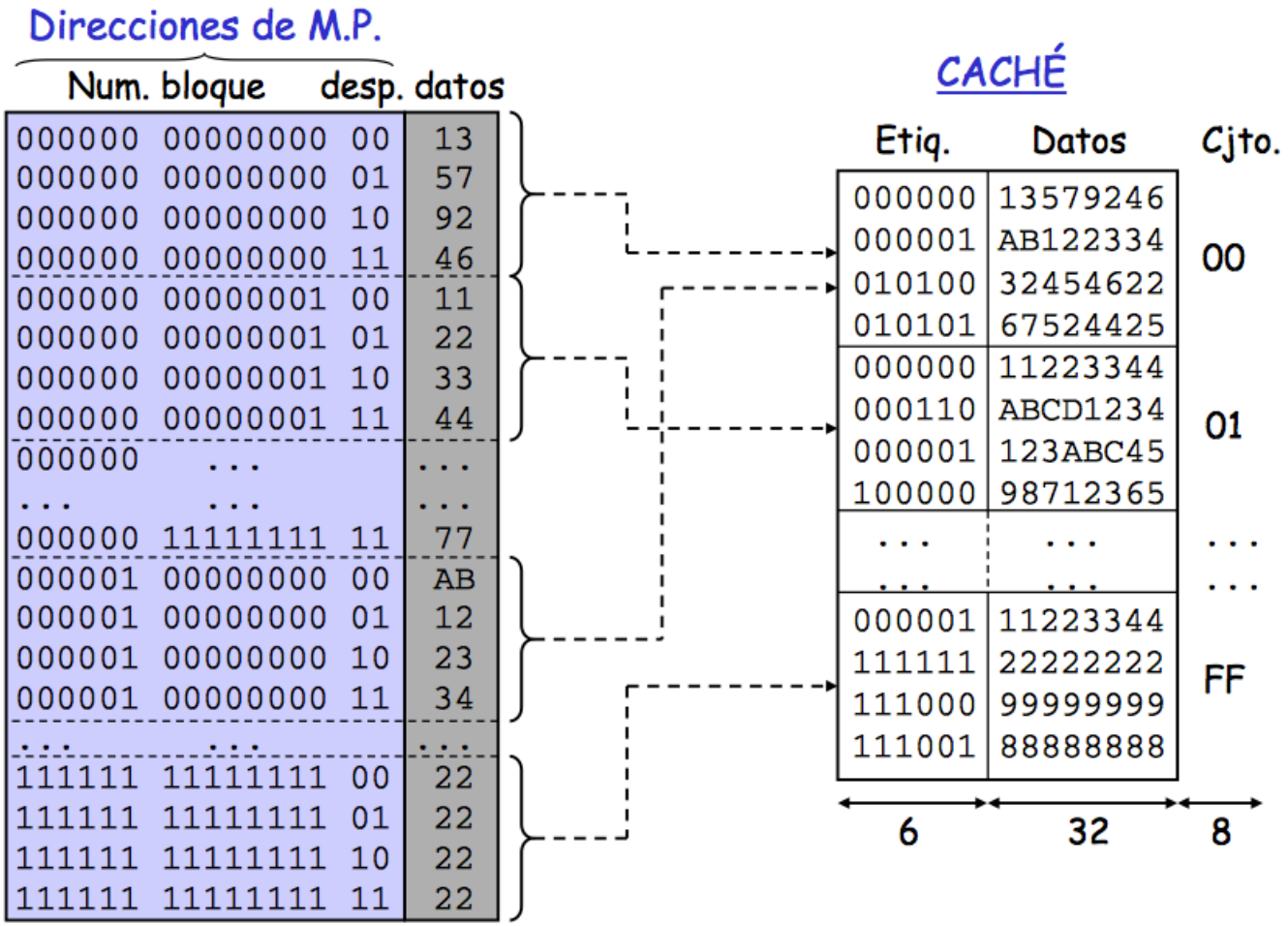
- Memoria principal: 64 Kbytes (16 Kbloques de 4 bytes)

Dirección M.P.

Número de bloque		despl.
6	8	2
etiqueta	conjunto	

Cjto. caché	Bloques de memoria principal
0	0, 100, 200, ..., 3F00
1	1, 101, 201, ..., 3F01
. . .	. . .
FF	FF, 1FF, 2FF, ..., 3FFF

CACHÉ ASOCIATIVA POR CONJUNTOS DE 2 VÍAS - Ejemplo



POLÍTICA DE REEMPLAZO DE BLOQUES

En una caché totalmente asociativa, hay que determinar:

¿qué bloque debe de ser eliminado cuando el conjunto se llena?

- Random
- Least Recently Used (LRU)
  - El estado de la cache LRU debe actualizarse con cada acceso.
  - Esta implementación sólo puede realizarse en conjuntos pequeños (2-way)
  - El arbol binario Pseudo-LRU se usa a menudo con 4-8 vias
- First In, First Out (FIFO) a.k.a. Round-Robin
  - Se usa en caches altamente asociativas.
- Least frequently used (LFU)

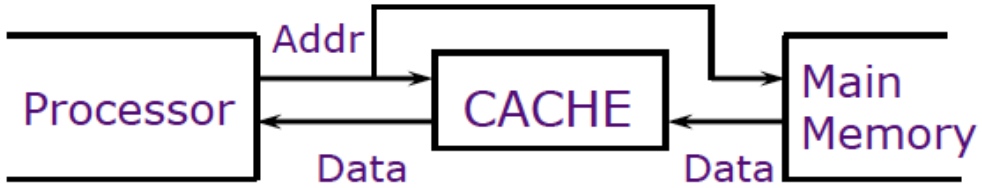
LATENCIA PROMEDIO DE LECTURA EN LA CACHÉ

α is HIT RATIO: Fraction of references in cache
1 - α is MISS RATIO: Remaining references

Average access time for serial search:

$t_c + (1 - \alpha)\, t_m$

Average access time for parallel search:



$\alpha\, t_c + (1 - \alpha)\, t_m$

Imagen de Emer, 2005

PRESTACIONES DE LA CACHÉ (POR INSTRUCCIÓN)

- tCPU=(NCPU+NMEM)xT
  - tCPU=CPU time [sec]
  - NCPU=CPU execution clock cycles [cycles]
  - NMEM =Memory stall clock cycles [cycles]
  - T=clock cycle time [sec]
- NMEM = NRxMRRxPR+ NWxMRWxPW
  - NMEM =Memory stall clock cycles [cycles]
  - NR/W= number of Reads/Writes [#]
  - MRR/W= Read/Write miss rate [fraction]
  - PR/W = Read/Write miss penalty [cycles]
- Combinando lectura y escritura.
- NMEM = NMxMRxP
  - NM=Memory accesses [#]
  - MR=Miss rate [fraction]
  - P= Miss penalty [cycles]

PRESTACIONES DE LA CACHÉ (PROGRAMA ENTERO, SE ASUME 1 INSTRUCCIÓN CADA VEZ)

- tCPU=ICx(CPIexecution+MPIxMRxP)xT

  IC=Instruction Count [#]

  tCPU=CPU time [sec]

  MPI=memory access per Instruction [#/instr]

- mPI=MPIxMR

  mPI= Misses per instruction [#/instruction]

- tCPU=IC x (CPIexecution +mPIxP)xT

- CPIexecution incluye los cache hit time (100% L1 hit rate)

MEJORA DE LAS PRESTACIONES DE LA CACHÉ

- Average memory access time = Hit time + Miss rate x Miss penalty.
- Para mejorar las prestaciones:
  - Reducir la tasa de fallo (miss rate) (e.g., cache mas grande).
  - Reducir la penalización por fallo (the miss penalty) (e.g., cache L2).
  - Reducir el tiempo de acierto (hit time).

REDUCCIÓN DEL TIEMPO DE ESCRITURA EN ACIERTO

- Problema: la escritura requiere dos ciclos en la etapa de memoria, un ciclo para verificar la etiqueta (tag check) mas un ciclo para la escritura del dato si hay acierto (hit).
- Soluciones:
  - Diseñar la RAM de datos para que pueda realizar las operaciones de lectura y escritura en un sólo ciclo, restaurando el valor antiguo después de un fallo de etiqueta (tag miss).
  - CAM-Tag caches: la línea de la palabra (word) sólo se valida si hay acierto (hit).
  - Escrituras segmentadas (pipelined writes): se mantiene el dato de escritura para almacenamiento en un buffer antes de la cache, se escribe los datos en cache durante la siguiente verificación de etiqueta para almacenamiento.

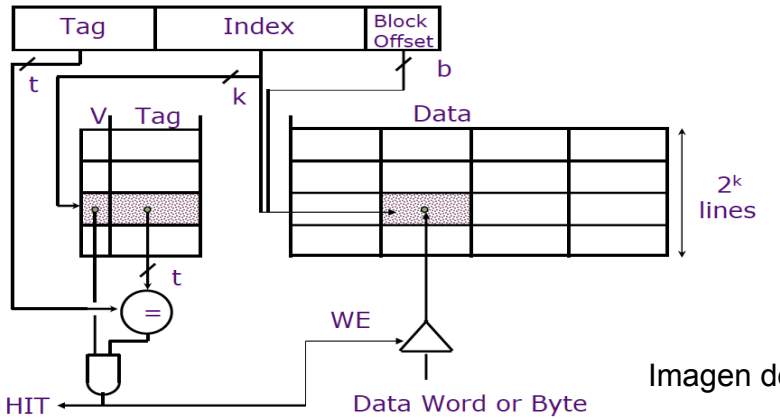


Imagen de Emer, 2005

POLÍTICA DE ESCRITURA

Hay dos aspectos a considerar cuando se ha de reemplazar:

- Cache hit (acierto):
  - write through (escritura inmediata): se escribe en cache y en memoria.
    - Generalmente tiene mayor tráfico pero simplifica el problema de coherencia de la caché.
  - write back (post escritura): se escribe la cache sólo (la memoria se escribe sólo cuando la entrada es eliminada de la caché).
    - Se usa un dirty bit por cada bloque para reducir aún más el tráfico.
    - Minimiza las escrituras en memoria principal, pero aparecen problemas coherencia sobre todo en sistemas multiprocesador.
- Cache miss (fallo):
  - no write allocate: sólo se escribe a la memoria principal.
  - write allocate (aka fetch on write): se escribe en memoria y se trae a la caché.
- Combinaciones comunes:
  - write through / no write allocate.
  - write back / write allocate.

REFERENCIAS

- [Emer, 2005] Joel Emer, Multilevel Memories, Computer System Architecture, OCW, http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-823-computer-system-architecture-fall-2005/
- [Aylagas, 2013] Memoria, Arquitectura de Computadores, UPM, http://www.dia.eui.upm.es/asignatu/Arq_com/AC%20Grado/Paco%20Aylagas/7-Memoria.pdf
- [Wikipedia, 2014] Magnetic-core memory, http://en.wikipedia.org/wiki/Magnetic-core_memory
- [Wikipedia, 2014b] Dynamic random access memory, http://en.wikipedia.org/wiki/Dynamic_random_access_memory