





Cache

Memoria Caché – Resumen Completo y Sencillo

¿Qué es la memoria caché?

La **caché** es una memoria rápida y pequeña que guarda datos que el procesador usa seguido. Sirve para **acelerar el acceso** a la información y evitar ir todo el tiempo a la memoria principal (RAM), que es más lenta.

¿Cómo funciona?

1. El procesador pide un dato.
2. Primero lo busca en la caché:
 - Si está →  **HIT** (acierto): se usa directo.
 - Si no está →  **MISS** (fallo): se busca en RAM o en niveles más profundos (L2, L3).
3. Si se encuentra, se copia a la caché para futuros accesos.

Niveles de caché

Nivel	Ubicación	Velocidad	Tamaño
L1	Dentro del núcleo del CPU	Muy rápida	Muy pequeña
L2	Entre CPU y RAM	Rápida	Mediana
L3	Compartida entre núcleos	Más lenta	Grande

Componentes clave

TAG (Etiqueta)

Identifica de qué parte de la memoria viene el dato. Se usa para comparar si el dato en caché es el que se necesita.

◆ SET INDEX

Indica en qué grupo (set) de la caché buscar el dato.

◆ OFFSET

Marca la posición exacta dentro del bloque de datos.

Bits importantes

Bit	Significado	¿Qué indica?
V (Valid)	Válido	Si el dato sirve o está vacío
D (Dirty)	Sucio	Si el dato fue modificado y aún no se guardó en la memoria principal

Asociatividad

Define **dónde puede ir un dato** en la caché:

- **Directa:** cada dato tiene un único lugar.
- **Totalmente asociativa:** puede ir en cualquier línea.
- **Por conjuntos (Set-Associative):** puede ir en cualquier línea dentro de su grupo.

LRU – *Least Recently Used*

Significa “el menos usado recientemente”. Es una **política de reemplazo** en caché.

¿Cómo funciona?

Cuando la caché está llena y hay que meter un dato nuevo, LRU **borra el dato que hace más tiempo que no se usa**.

Ventaja:

- Mantiene los datos más “calientes” (usados seguido).

Desventaja:

- Hay que llevar registro del uso de cada dato → más complejidad.



Thumb – *Modo reducido de ARM*

Parece que lo que viste en tus apuntes es **Thumb**, no "Trum". Es un **modo de ejecución** en procesadores ARM.



¿Qué hace Thumb?

- Usa **instrucciones de 16 bits** en vez de 32.
- Ocupa menos espacio en memoria.
- Ideal para sistemas con poca RAM o que necesitan ahorrar energía.

Ejemplo criollo:

Es como usar abreviaturas al escribir. Hace lo mismo pero ocupa menos

Ventaja:

- Ahorra memoria.
- Permite meter más instrucciones por bloque.

Desventaja:

- Tiene menos funciones que el modo completo.



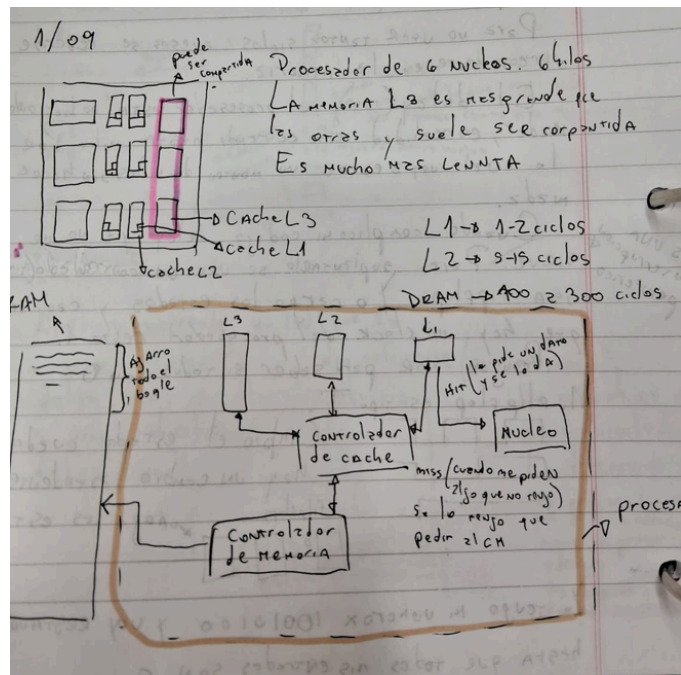
Ejemplo práctico

Direcciones: `0x20`, `0x24`, `0x28`, `0x2C`

Si tenés 4 sets, cada dirección cae en un set distinto.

Si accedés varias veces y el dato ya está en caché → HIT.

Si no está → MISS → se copia y se marca como válido.



- **Asociatividad baja (1 o 2):**
 - Más rápida.
 - Pero más chances de **conflictos** (dos datos que quieren el mismo lugar).
 - Más **misses** si se repiten direcciones.
- **Asociatividad alta (4 o más):**
 - Más flexible.
 - Menos conflictos.
 - Pero más lenta para buscar (más comparaciones).

Ejemplo práctico

Supongamos que tenés una caché con 4 sets y asociatividad 2:

- Cada set tiene 2 líneas → cada dato puede ir en **2 lugares posibles** dentro de su set.
- Si llegan 3 datos que caen en el mismo set → uno se tiene que **reemplazar** (usás LRU, FIFO, etc.).

Cache Replacement Policies (LRU, Tree-pLRU, MRU, QLRU, FIFO, LFU, Random and more).