PROGRAMADOR UNIVERSITARIO
LICENCIATURA EN INFORMÁTICA
INGENIERÍA EN INFORMÁTICA
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II
TRABAJO PRÁCTICO Nº 5
PRADO, MATIAS SANTIAGO

1) ¿Cuántos bits en total se requieren para una caché de mapeo directo con 8 KB de datos y bloques de 8 palabras de 32 bits cada una, asumiendo una dirección de 64 bits? Detalle cuántos bits para el índice, la etiqueta, datos y validación.
Datos: 8KB = $8 * 8 * 2^{10} = 65536\ bits$
Tamaño bloque: 8 palabras
Palabra: 32 bits
Dirección: 64 bits

Como tenemos 8 palabras por bloque (cada linea) y cada palabra tiene 32bit tenemos

$$tamanio\ bloque = 8 * 4 = 32\ bytes = 256\ bits$$

Luego el numero de bloques será

$$numero\ lineas = \frac{65536}{256} = 256\ lineas$$

Luego el numero de bits de offset será

$$log_2(32) = 5\ bits$$

El numero de bits para el índice será

$$log_2(256) = 8\ bits$$

Ahora restando de la dirección el numero de bits del indice y offset tendremo el numero de bit de la tag

$$64 - 5 - 8 = 51\ bits \rightarrow tag$$

Por lo tanto tenemos:
Offset = 5 bits
Índice = 8 bits
Tag = 51 bits
Datos =256 bits (bits de palabras por línea)
Validación = 1bit

2) ¿Qué significa que una memoria caché sea totalmente asociativa? Supongamos que tenemos una caché con 8 conjuntos y 4 líneas por conjunto. ¿Cómo se mapearían las direcciones de memoria en esta caché? ¿Cuál sería la función de búsqueda en esta estructura?

Una cache totalmente asociativa a diferencia de una de mapeo directo donde cada línea de cache puede corresponder a n líneas de memoria, mientras que una linea en una cache asociativa podemos tener multiples líneas de memoria para el mismo índice, como la comparación siguiente brindada por Gemini AI:

Caché de Mapeo Directo

- Tienes un armario con 256 cajones (256 **índices**).
- Cada cajón (índice) tiene **UN SOLO ESPACIO** (es de 1 vía).
- En ese único espacio, guardas tu bloque de datos (ej: 8 palabras).
- **Problema:** Si llega un bloque de memoria que va en el "índice 5", pero ya tenías un bloque en el "índice 5", estás obligado a **tirar el bloque viejo** y poner el nuevo. Hay un conflicto.

Caché Asociativa por Conjuntos (Ej: 4 vías)

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II
TRABAJO PRÁCTICO Nº 5
PRADO, MATIAS SANTIAGO

- Tienes un armario con 8 estantes (8 **índices**, que ahora llamamos **conjuntos**).
- Cada estante (conjunto) tiene **CUATRO ESPACIOS** (es de 4 **vías**).
- En cada uno de esos 4 espacios, guardas un bloque de datos completo (ej: 8 palabras).
- **Ventaja:** Si llega un bloque de memoria que va en el "índice 5" (Conjunto 5), miras el estante.
    - ¿El espacio 1 (vía A) está libre? Lo pones ahí.
    - ¿Está ocupado? Miras el espacio 2 (vía B). ¿Está libre? Lo pones ahí.
    - Solo tienes un conflicto si los 4 espacios (las 4 vías) de ese estante (conjunto) ya están llenos con datos que también mapean al "índice 5".

3) Una computadora de 32 bits con una memoria caché de 256 KB y líneas de 64 bytes. La caché es asociativa por conjuntos de 4 vías. Se pide: a) Indique el número de líneas y de conjuntos de la memoria caché del enunciado. b) ¿Cuál es el tamaño de los bloques que se transfieren entre la memoria caché y la memoria principal?

$tamaño\ palabra = 32\ [bit]$
$tamaño\ cache = 256\ [KB]$
$tamaño\ linea = 64\ [Byte]$
$N_{vias} = 4$

a)

$$N_{lineas} = \frac{tamaño\ cache}{tamaño\ linea} = \frac{256 * 2^{10}}{64} = 4096\ [lineas]$$

$$N_{conjuntos} = \frac{N_{lineas}}{N_{vias}} = \frac{4096}{4} = 1024\ [conjuntos]$$

b)

El tamaño mínimo de bloque que se transfiere entre memoria y cache es de 64 bytes, puesto que esto es lo que mide una linea/bloque de cache.

PROGRAMADOR UNIVERSITARIO
LICENCIATURA EN INFORMÁTICA
INGENIERÍA EN INFORMÁTICA
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II
TRABAJO PRÁCTICO Nº 5
PRADO, MATIAS SANTIAGO

4) Se dispone de una computadora con direcciones de memoria de 32 bits, que direcciona la memoria por bytes. La computadora dispone de una memoria caché asociativa por conjuntos de 4 vías, con un tamaño de línea de 4 palabras. Dicha caché tiene un tamaño de 64 KB. Indique de forma razonada:

a) Tamaño en MB de la memoria que se puede direccionar en este computador.
b) Número de palabras que se pueden almacenar en la memoria caché.
c) Número de líneas de la caché.
d) Número de conjuntos de la caché.

a)

$T_{palabra} = 32\ [bit] = 4\ [Byte]$
$T_{linea} = 4\ palabra = 16\ [Byte]$
$T_{cache} = 64\ [KB]$
$N_{vias} = 4$

$$Total_{memoria} = 2^{32}\ [Byte] = 2^{12} = 2^2 * 2^{10} = 4096\ [MB]$$

b) y c)

$$N_{lineas\ cache} = \frac{T_{cache}}{T_{linea}} = \frac{64 * 2^{10}}{16} = 4096\ [lineas]$$

ahora como por cada líneas tenemos 4 palabras entonces la cantidad total de palabras que puede almacenar la cache será:

$$Total_{palabras\ cache} = N_{lineas\ cache} * T_{linea} = 4096 * 4 = 16384\ [palabras]$$

d)

$$N_{conjuntos} = \frac{N_{lineas\ cache}}{N_{vias}} = \frac{4096}{4} = 1024\ [conjuntos]$$

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II
TRABAJO PRÁCTICO Nº 5
PRADO, MATIAS SANTIAGO

5) De acuerdo a un estudio realizado sobre la utilización de las instrucciones de una computadora RISC, en promedio ejecuta 50 millones de instrucciones por segundo y que el porcentaje de utilización de sus instrucciones es el siguiente:

- LOAD un 30 %
- STORE un 15 %
- Operaciones aritméticas un 29 %
- Operaciones lógicas un 6 %
- Saltos un 20 %

Teniendo que cuenta que el acceso a las instrucciones cuentan como accesos a la memoria puesto que se tiene que hacer un fetch para traerlas, de base tenemos 50.000.000 de instrucciones, luego tenemos sobre esas un 45% mas (30% LOAD y 15% STORE) que son las que tambien requieren acceso a memoria, entonces en total tenemos

$$N_{accesos} = 50M + 50M * 0.45 = 72500000 \left[\frac{accesos}{s}\right]$$

6) Basados en el ejemplo de la teoría, determinar en hay Hit o Miss en cada referencia de la secuencia y completar el bloque de caché desde su estado inicial

Referencia de dirección (decimal)	Referencia de dirección (binario)	Hit o Miss en Caché	Bloque de cache asignado
3			
3			
21			
7			
21			
27			
5			
7			
24			

Ref. Dir. Decimal	Ref. Dir. Binario	Hit o Miss en cache	Bloque de cache asignado
3	00011	Miss	011
3	00011	Hit	011
21	10101	Miss	101
7	00111	Miss	111
21	10101	Hit	101
27	11011	Miss	011
5	00101	Miss	101
7	00111	Hit	111
24	11000	Miss	

Estado inicial de la cache

Índice	V	Tag	Data
000	N	11	Memory (11000)
001	N		
010	N		
011	V	11	Memory (11011)
100	N		
101	V	00	Memory (00101)
110	N		
111	V	00	Memory(00111)

7) Describe la política de reemplazo LRU (Least Recently Used) y cómo se utiliza en la memoria caché

Mantiene una lista ordenada: El algoritmo mantiene una lista ordenada de cada línea (entrada) dentro de un conjunto.

Actualización en cada acceso: Cada vez que se accede a una de las líneas que ya están presentes en el conjunto (un *hit*), la lista se actualiza para marcar esa línea como la "entrada a la que se accedió más recientemente".

Momento del reemplazo: Cuando ocurre un fallo (*miss*) y se debe traer un nuevo bloque a un conjunto que ya está lleno, el algoritmo elije para descartar a la línea que está al final de la lista, es decir, aquella a la que se accedió "menos recientemente"

8)
Explica las diferencias entre la escritura en caché por escritura inmediata (write-through) y escritura en caché por escritura diferida (write-back). ¿Cuáles son las ventajas y desventajas de cada enfoque de escritura en caché?

Escritura Directa (Write-Through)

- **Cómo funciona:** En este esquema, cuando la CPU escribe un dato, la escritura se realiza **siempre tanto en la caché como en la memoria principal**. Esto mantiene la memoria principal y la caché consistentes.
- **Ventajas:**
  - Es el **diseño más sencillo**.
  - Garantiza que la memoria principal esté siempre actualizada (consistente) con la caché.
- **Desventajas:**
  - Proporciona un **rendimiento muy malo**.
  - Cada escritura obliga al procesador a esperar los tiempos lentos de la memoria principal, lo que puede ralentizar considerablemente la ejecución.
  - Se puede mitigar parcialmente esta desventaja usando un **búfer de escritura**.

Escritura Diferida (Write-Back)

- **Cómo funciona:** En este esquema alternativo, cuando la CPU realiza una escritura, el nuevo valor se escribe **solo en el bloque de la caché**.
- El bloque modificado se escribe en la memoria principal (un nivel inferior) **SOLO cuando dicho bloque va a ser reemplazado** de la caché.
- **Ventajas:**
  - **Mejora el rendimiento**, ya que el procesador no tiene que esperar a la memoria principal en cada escritura.
- **Desventajas:**
  - Es un esquema **más complejo de implementar**.

9) Describe los conceptos de localidad temporal y localidad espacial en el contexto de la memoria caché. Explica cómo la localidad puede afectar el rendimiento de la memoria caché.

1. Localidad Temporal

Este concepto (que no se define explícitamente en el PDF de teoría, pero es fundamental) se refiere a que si un programa accede a un dato o instrucción, es **muy probable que vuelva a acceder a ese mismo dato o instrucción** en un futuro cercano.

- **Cómo se usa en la Caché:** La caché guarda los datos usados recientemente. Gracias a la localidad temporal, la próxima vez que la CPU pida ese mismo dato, lo encontrará en la caché (un *hit*), lo cual es mucho más rápido.

2. Localidad Espacial

La teoría sí define esta localidad. Se refiere a que si un programa accede a un dato en una dirección de memoria, es **muy probable que pronto acceda a otras direcciones cercanas**.

- **Cómo se usa en la Caché:** Cuando ocurre un *miss*, la caché no trae solo la palabra solicitada; trae el **bloque completo** (la línea de caché) que la rodea. Esto aprovecha la localidad espacial al

"pre-cargar" los datos cercanos, anticipando que serán los próximos en ser pedidos, lo que reduce las tasas de fallas futuras.

¿Cómo afecta esto al rendimiento?

La localidad es la razón de ser de la caché.

- Un **alto grado de localidad** (temporal y espacial) en un programa significa que la mayoría de los accesos de la CPU serán **aciertos (hits)** en la caché.
- Cada *hit* es un acceso rápido. Cada *miss* (fallo) es una penalización de tiempo, ya que se debe ir a la memoria principal lenta.

Por lo tanto, un programa con buena localidad tendrá un rendimiento excelente porque la caché puede satisfacer la gran mayoría de sus peticiones de forma casi instantánea