DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (II/2025)

Ayudantía 12: Jerarquía de Memoria

Ayudantes: Daniela Ríos (danielaarp@uc.cl), Alberto Maturana (alberto.maturana@uc.cl), Fernanda Escobar (fer_jez2002@uc.cl)

Pregunta 1: Evaluación de una Jerarquía de Memoria

En la jerarquía de memoria de un computador, cuenta con una memoria caché de 128KiB, líneas de 32 palabras y tiempo de acceso de 8ns. Por otra parte, se sabe que el tiempo que toma, luego de un *miss*, copiar un bloque de memoria del siguiente nivel de la jerarquía en la caché, y luego acceder al dato buscado desde esta es de 80ns . Si se realizan 10 accesos a memoria, cuantos *hits* deben ocurrir para obtener un tiempo de acceso promedio de 16ns en el computador?

Solución: Para obtener el tiempo de acceso promedio en una memoria, se utiliza la siguiente fórmula:

$$TP = HR \times HT + (1 - HR) \times (HT + MP)$$

Donde TP es el tiempo de acceso promedio; HT el *hit-time* promedio; HR el *hit-rate* promedio; y MP el *miss penalty*.
Del enunciado se sabe que el *hit-time* promedio es 8ns y el tiempo de acceso promedio es 16ns. Asimismo, nos dicen que el tiempo que toma, luego de un *miss*, copiar un bloque de memoria del siguiente nivel de la jerarquía en la caché, y luego acceder al dato buscado desde esta es de 80ns, es decir, el *miss penalty* es de 80ns. Luego, reemplazando valores tenemos que:

$$16\text{ns} = HR \times 8\text{ns} + (1 - HR) \times (8\text{ns} + 80\text{ns})$$

$$16\text{ns} = 88\text{ns} - 80\text{ns} \times HR$$

$$HR = \frac{72\text{ns}}{80\text{ns}}$$

$$HR = 0{,}9$$

Una vez obtenido el *hit-rate*, para calcular la cantidad de *hits* reemplazamos en la siguiente fórmula:

$$HR = \frac{\#Hits}{\#Accesos}$$

$$0,9 = \frac{\#Hits}{10}$$

$$\#Hits = 9$$

Pregunta 2: Jerarquía de Memoria (P3-I2-2023-1)

(a) Suponga que, en un instante dado, posee el siguiente estado una caché de 8 líneas y 2 palabras por línea, cada una de 1 byte:

Línea	Validez	*Timestamp*	*Tag*
0	1	2	5
1	1	7	15
2	0	5	9
3	1	0	1
4	1	3	3
5	1	6	9
6	1	4	5
7	1	1	8

La columna "Línea" representa el índice de línea; "Validez" representa el *valid bit*; "*Timestamp*" representa el tiempo **desde el último acceso a la línea**; y "*Tag*" representa el valor, en base decimal, del *tag* que indica el bloque de memoria almacenado en la línea. Con este estado de caché, se realiza el acceso a memoria a las siguientes direcciones: 0x8F, 0x17, 0x11, 0x3D, 0xF5, 0x16. Indique, para cada acceso, si existe un *hit* o *miss*; si corresponde, la línea de la caché que es modificada; y el *hit-rate* para dos funciones de correspondencia distintas:

1. (3 ptos.) *Directly mapped*.
2. (3 ptos.) *4-way associative*.

Asuma que posee una memoria principal de 256 palabras de 1 byte y que la política de reemplazo utilizada es LRU, si corresponde (*i.e.* se reemplaza la línea con mayor *timestamp*). Para responder, complete las tablas adjuntas al enunciado. El *tag* de cada línea lo puede escribir en base binaria o decimal. No necesita indicar la actualización de *timestamp*, pero sí debe considerar su valor en caso de reemplazos.

Solución: Independiente de la función de correspondencia, al tener una memoria de 256 palabras se necesitan $\log_2(256) = 8$ bits para cada dirección. Por otra parte, al ser las líneas de 2 palabras, se requiere $\log_2(2) = 1$ bit de *offset* para ubicar una palabra dentro de una línea. Con esto en consideración, se tiene el siguiente resultado para cada secuencia de accesos según función:

1. *Directly mapped*: Al tener 8 líneas, se necesitan $\log_2(8) = 3$ bits de índice de línea, resultando en $8 - 3 - 1 = 4$ bits de *tag*. Por otra parte, el *timestamp* es irrelevante ya que no aplican las políticas de reemplazo en esta función, si el contenido no está en la línea a la que pertenece, se reemplaza directamente. A continuación, la tabla de accesos resultante:

N° Acceso	Dirección	*Tag*	*Hit/Miss*	ID Línea modificada	Comentarios
0	0x8F	1000b = 8 = 0x8	*Hit*	-	Índice de línea igual a 111b = 7. El bit de validez es 1 y coinciden los *tags*.
1	0x17	0001b = 1 = 0x1	*Hit*	-	Índice de línea igual a 011b = 3. El bit de validez es 1 y coinciden los *tags*.
2	0x11	0001b = 1 = 0x1	*Miss*	000b = 0	Índice de línea igual a 000b = 0. El bit de validez es 1 pero no coinciden los *tags*.
3	0x3D	0011b = 3 = 0x3	*Miss*	110b = 6	Índice de línea igual a 110b = 6. El bit de validez es 1 pero no coinciden los *tags*.
4	0xF5	1111b = 15 = 0xF	*Miss*	010b = 2	Índice de línea igual a 010b = 2. El bit de validez es 0, por lo que el contenido buscado no está.
5	0x16	0001b = 1 = 0x1	*Hit*	-	Índice de línea igual a 011b = 3. El bit de validez es 1 y coinciden los *tags*.

Hit-rate: $\frac{3}{6} = 0{,}5 = 50\,\%$

2. *4-way associative*: Al tener conjuntos de 4 líneas, se tienen $\frac{8}{4} = 2$ conjuntos y, de esta forma, $\log_2(2) = 1$ bit de índice de conjunto, resultando en $8 - 1 - 1 = 6$ bits de *tag*. En este caso, el conjunto 0 corresponde a las líneas 0-3 y el conjunto 1 a las líneas 4-7. A continuación, la tabla de accesos resultante:

N° Acceso	Dirección	*Tag*	*Hit/Miss*	ID Línea modificada	Comentarios
0	0x8F	100011b = 35 = 0x23	*Miss*	101b = 5	Índice de conjunto igual a 1. No existe coincidencia de *tag*. Se reemplaza la línea de mayor *timestamp*, en este caso, la línea 5. La nueva línea de mayor *timestamp* en el conjunto es la 6.
1	0x17	000101b = 5 = 0x5	*Hit*	-	Índice de conjunto igual a 1. El *tag* coincide con el de la línea 6 y su bit de validez es 1, por lo que hay *hit*. La nueva línea de mayor *timestamp* en el conjunto es la 4.
2	0x11	000100b = 4 = 0x4	*Miss*	010b = 2	Índice de conjunto igual a 0. No existe coincidencia de *tag*. El contenido se escribe en la única línea disponible, que corresponde a la 2. La línea de mayor *timestamp* en el conjunto sigue siendo la 1.
3	0x3D	001111b = 15 = 0xF	*Hit*	-	Índice de conjunto igual a 0. El *tag* coincide con el de la línea 1 y su bit de validez es 1, por lo que hay *hit*. La nueva línea de mayor *timestamp* en el conjunto es la 0.
4	0xF5	111101b = 61 = 0x3D	*Miss*	000b = 0	Índice de conjunto igual a 0. No existen coincidencia de *tag*. Se reemplaza la línea de mayor *timestamp*, en este caso, la línea 0. La nueva línea de mayor *timestamp* en el conjunto es la 3.
5	0x16	000101b = 5 = 0x5	*Hit*	-	Índice de conjunto igual a 1. El *tag* coincide con el de la línea 6 y su bit de validez es 1, por lo que hay *hit*. La línea de mayor *timestamp* en el conjunto sigue siendo la 4.

Hit-rate: $\frac{3}{6} = 0{,}5 = 50\,\%$

Para cada función de correspondencia, se asignan **0.5 ptos.** por acceso correctamente descrito (*i.e.* obtención de *tag*; determinación de *hit* o *miss* y selección de línea a reemplazar). Si existen errores de arraste, se otorgan como máximo **1.5 ptos.** por función si en el desarrollo se evidencia una correcta obtención de bits de *offset*, índice y *tag* por dirección.

(b) Suponga que posee un espacio de direcciones virtuales de 20 bits, direcciones físicas de 19 bits y un tamaño de páginas de 128 KiB. En este esquema, un proceso P_i posee el siguiente

estado para su tabla de páginas en un instante dado:

Entrada	Validez	Marco
0	0	Disco
1	0	1
2	1	0
3	0	2
4	0	2
5	1	3
6	0	Disco
7	1	1

La columna "Entrada" representa el número de entrada de la tabla; la columna "Validez" representa si el contenido del marco físico es válido; y "Marco" representa el valor, en base decimal, del marco físico asociado a la página ligada a la entrada. Si en esta celda se encuentra el valor "Disco", esto indica que su contenido se encuentra en el *swap file*. Con este estado de tabla de páginas, se realiza el acceso a memoria a las siguientes direcciones virtuales: 0x8F3A0, 0x54168, 0xE07C4, 0x1701A. Indique, para cada acceso, si existe un *page fault* o no. Solo en caso de no existir *page fault* (*i.e.* que la página de la dirección virtual posea un marco físico asociado válido), indique el número de marco y la dirección física resultante. Para responder, se solicita que complete la tabla adjunta al enunciado. Puede escribir la dirección física en base binaria o decimal.

Solución: Al tener un tamaño de páginas de 128 KiB, se tiene un total de $2^7 \times 2^{10} = 2^{17}$ palabras de memoria por página, por lo que se tienen $\log_2(2^{17}) = 17$ bits de *offset*. De esto se deduce que el número de marco físicos es de $19 - 17 = 2$ bits y el número de página es de $20 - 17 = 3$ bits. Este último coincide con las dimensiones de la tabla de páginas, dado que se tienen $2^3 = 8$ entradas. A continuación, la tabla de traducciones resultante:

N° Acceso	Dirección virtual	Número página	¿*Page fault*?	Número marco	Dirección física	Comentarios
0	0x8F3A0	100b = 4	Sí	-	-	*Page fault* por bit de validez.
1	0x54168	010b = 2	No	00b = 0	00b|1b|0x4168 = 00101000001011010000b = 82280 = 0x14168	
2	0xE07C4	111b = 7	No	01b = 1	01b|0b|0x07C4 = 01000000111110001000b = 133060 = 0x207C4	
3	0x1701A	000b = 0	Sí	-	-	*Page fault* por marco en disco.

Se asignan **0.5 ptos.** por acceso correctamente realizado (*i.e.* determinación de *page fault* o traducción de dirección virtual a física). Se acepta expresar la dirección física resultante como la concatenación de bits y dígitos hexadecimales. Si existen errores de arrastre, se otorga como máximo **1 pto.** si en el desarrollo se evidencia una correcta obtención de bits de *offset*, número de página y número de marco físico para cada dirección virtual.

Pregunta 3: Jerarquía de Memoria (P1-I3-2025-1)

(a) Indique y justifique qué principio de localidad es explotado con la lógica de bloques de memoria y líneas de caché en una jerarquía de memoria.

Solución: La lógica de bloques de memoria y líneas de caché explota el principio de localidad espacial. Este principio señala que es "probable que datos cercanos al accedido también sean utilizados", por lo que se almacena en caché el contenido del bloque contiguo al que pertenece un dato y no solo este de forma individual.

(b) Para estudiar el comportamiento de una caché N-way associative de L líneas a partir del valor de N, evalúa dos casos: (1)N=1; (2)N=L. Para cada caso, explique cómo se comporta el mapeo de un bloque de memoria a una línea de caché.Si se asemeja al comportamiento de otra función de correspondencia conocida, indíquela y justifique por qué.

Solución:
(1) Si $N = 1$, cada conjunto es de una línea. Esto implica que existen tantos conjuntos como líneas haya en la caché. Como cada bloque se puede mapear a un solo conjunto, esto es equivalente a que se pueda mapear a una sola línea. Por lo tanto, su comportamiento se asemeja al de la función de correspondencia *directly mapped*.
(2) Si $N = L$, cada conjunto es de L líneas. Esto implica que existe un solo conjunto que alberga todas las líneas de la caché. Como cada bloque se puede mapear a cualquier línea dentro de un conjunto, esto es equivalente a que se pueda mapear a cualquier línea de la caché. Por lo tanto, su comportamiento se asemeja al de la función de correspondencia *fully associative*.

(c) Suponga que, para una memoria principal de 1024 palabras de 1 byte, posee una caché de 8 líneas y 16 palabras por línea con el siguiente estado intermedio:

Línea	*Valid*	*Tag*	*Time*
0	1	57	5
1	1	27	10
2	1	40	2
3	1	34	8

Línea	*Valid*	*Tag*	*Time*
4	1	58	7
5	1	9	4
6	1	62	3
7	1	0	1

La columna "Línea" representa el índice de línea; "Validez" representa el *valid bit*; "Tag" representa el valor decimal del *tag* que indica el bloque de memoria almacenado en la línea; y "Timestamp" representa el tiempo desde el último acceso a la línea (*i.e.* **un valor menor representa un acceso más reciente**). A partir de este estado, se realiza el acceso a memoria de las siguientes direcciones: `0x3AA`, `0x1B0`, `0x00F`, `0x1FF`, `0x01A` y `0x2BB`. Indique, para cada acceso, si hay *hit* o *miss*; la línea de caché accedida o modificada; y el *hit-rate* para la función de correspondencia ***fully associative*** **con política de reemplazo LRU.** Para responder, complete las tablas adjuntas al enunciado. El *tag* de cada línea lo puede escribir en base binaria o decimal. No necesita indicar el *timestamp* final por línea, pero sí debe usarlo en caso de reemplazos. **Considere que, si bien las direcciones se dan en hexadecimal, su**

representación binaria no necesariamente representa la cantidad de bits de cada dirección de memoria. Esta cantidad se debe deducir de la información otorgada.

Solución:
Primero calculamos los bits necesarios para cada dirección. Como tenemos una memoria de 1024 palabras de 1 byte, se necesitan $log_2(1024) = 10$ bits para cada dirección.
Por otra parte, como la caché tiene 16 palabras por línea se necesitan $log_2(16) = 4$ bits de *offset*. Al tratarse de una caché *fully associative* la dirección está compuesta por el *tag* y el *offset* respectivamente, por lo tanto si tenemos 10 bits de dirección y 4 bits de *offset*, los 6 bits restantes son para el *tag*.

Una vez calculado esto, expresamos cada dirección en binario identificando su tag.

Dir (hexa)	Dir (bin)	*Tag*
0x3AA	1110101010b	$111010\text{b} = 58_{10}$
0x1B0	0110110000b	$011011\text{b} = 27_{10}$
0x00F	0000001111b	$000000\text{b} = 0_{10}$
0x1FF	0111111111b	$011111\text{b} = 31_{10}$
0x01A	0000011010b	$000001\text{b} = 1_{10}$
0x2BB	1010111011b	$101011\text{b} = 43_{10}$

Con esto en consideración se obtiene el siguiente resultado de la secuencia de accesos a memoria. Recordar que como estamos usando la política LRU, el reemplazo, en caso de existir, se efectuará en la línea con mayor tiempo sin acceso, es decir, aquella con mayor *timestamp*.

1. Acceso a la dirección 0x3AA. Resulta en *hit*, ya que la línea 4 posee *tag* igual a 58. La línea 1 sigue siendo la que posee mayor *timestamp*.

Línea	*Valid*	*Tag*	*Time*
0	1	57	6
1	1	27	11
2	1	40	3
3	1	34	9

Línea	*Valid*	*Tag*	*Time*
4	1	58	0
5	1	9	5
6	1	62	4
7	1	0	2

2. Acceso a la dirección 0x1B0. Resulta en *hit*, ya que la línea 1 posee *tag* igual a 27. Ahora la línea 3 es la que posee mayor *timestamp*.

Línea	*Valid*	*Tag*	*Time*
0	1	57	7
1	1	27	0
2	1	40	4
3	1	34	10

Línea	*Valid*	*Tag*	*Time*
4	1	58	1
5	1	9	6
6	1	62	5
7	1	0	3

3. Acceso a la dirección 0x00F. Resulta en *hit*, ya que la línea 7 posee *tag* igual a 0. La línea 3 sigue siendo la que posee mayor *timestamp*.

Línea	*Valid*	*Tag*	*Time*
0	1	57	8
1	1	27	1
2	1	40	5
3	1	34	11

Línea	*Valid*	*Tag*	*Time*
4	1	58	2
5	1	9	7
6	1	62	6
7	1	0	0

4. Acceso a la dirección 0x1FF. Resulta en *miss*, ya que no existe una linea válida con *tag* igual a 31. Se copia el bloque sobre la línea 3, que es la que posee mayor *timestamp*. Ahora la línea 0, que es la que posee mayor *timestamp*.

Línea	*Valid*	*Tag*	*Time*
0	1	57	9
1	1	27	2
2	1	40	6
3	1	31	0

Línea	*Valid*	*Tag*	*Time*
4	1	58	3
5	1	9	8
6	1	62	7
7	1	0	1

5. Acceso a la dirección 0x01A. Resulta en *miss*, ya que no existe una línea válida con *tag* igual a 1. Se copia el bloque sobre la línea 0, que es la que posee mayor *timestamp*. Ahora la línea 5, que es la que posee mayor *timestamp*.

Línea	*Valid*	*Tag*	*Time*
0	1	1	0
1	1	27	3
2	1	40	7
3	1	31	1

Línea	*Valid*	*Tag*	*Time*
4	1	58	4
5	1	9	9
6	1	62	8
7	1	0	2

6. Acceso a la dirección 0x2BB. Resulta en *miss*, ya que no existe una línea válida con *tag* igual a 43. Se copia el bloque sobre la línea 5, que es la que posee mayor *timestamp*. Ahora la línea 6, que es la que posee mayor *timestamp*.

Línea	*Valid*	*Tag*	*Time*
0	1	1	1
1	1	27	4
2	1	40	8
3	1	31	2

Línea	*Valid*	*Tag*	*Time*
4	1	58	5
5	1	43	0
6	1	62	9
7	1	0	3

Finalmente calculamos el *hit-rate*:

$$HR = \frac{\#Hits}{\#Accesos} = \frac{3}{6} = 0{,}5 = 50\,\%$$

Feedback ayudantía

Escanee el QR para entregar feedback sobre la ayudantía.