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

IIC2343 - Arquitectura de Computadores (II/2026)

Ayudantía 12

Ayudantes: Ignacio Gajardo (gajardo.ignacio@uc.cl), Mario Rojas (mario.denzel@estudiante.uc.cl), Nicolás Romo Aubele (nroma@uc.cl), Cristina Antón (ceanton@uc.cl)

Pregunta 1: Teoría

a) Usted trabaja en el diseño de la memoria caché del computador de su empresa. Esta caché es significativamente pequeña y rápida, por lo que los tiempos de búsqueda en ella son bajos. Su supervisor le solicita que, con el fin de usar de forma óptima la caché, implemente la política de reemplazo *Bélády*.

1. Según lo estudiado en el curso, ¿cómo respondería ante esta solicitud? Justifique su respuesta.
2. Para esta caché, ¿qué función de correspondencia recomendaría para optimizar el uso de su espacio? Justifique su respuesta.

b) Suponga que posee dos computadores con arquitecturas similares, salvo por sus memorias caché: poseen las mismas dimensiones, pero una es *16-way associative* y la otra es *8-way associative*. ¿Cuál presenta un *hit-time* promedio menor? Justifique su respuesta.

Solución:

a.1) La solicitud debe responderse de forma negativa. La política *Bélády* reemplaza el bloque que se utilizará más lejos en el futuro, lo que implica conocer los accesos futuros antes de que ocurran. Como no es posible ver el futuro, es un ideal inalcanzable en la práctica.

a.2) Conviene usar *fully associative*. Como la caché es pequeña y rápida, el costo de buscar el *tag* en todas las líneas es bajo, y es preferible aprovechar al máximo el espacio, como cualquier bloque puede ocupar cualquier línea, lo que maximiza el *hit-rate*. Un ejemplo es la TLB, que es *fully associative* por su tamaño y rapidez.

b) Considerando que ambas memorias caché poseen las mismas dimensiones, la que presenta un menor *hit-time* promedio corresponde a la *8-way associative*. Esto se debe a que la búsqueda en memorias caché *N-way associative* se realiza solo sobre las líneas del conjunto al que se mapea el bloque de la palabra buscada. Como N equivale a la cantidad de líneas por conjunto, en una caché *8-way associative* la búsqueda se realiza sobre la mitad de las líneas de una *16-way associative*, lo que implica un menor tiempo para determinar si hay *hit* o no.

Pregunta 2: P1.C 2026-1

Suponga que, para una memoria principal de 2048 palabras de 1 byte, posee una caché de 16 líneas y 16 palabras por línea con el siguiente estado intermedio:

Línea	*Valid*	*Tag*	*Time*
0	1	4	8
1	1	7	9
2	1	6	14
3	0	5	-
4	1	2	7
5	1	15	6
6	1	6	11
7	1	4	10

Línea	*Valid*	*Tag*	*Time*
8	1	15	12
9	1	8	13
10	1	12	5
11	1	4	4
12	1	4	2
13	1	11	3
14	1	3	1
15	0	1	-

La columna "Línea" representa el índice de línea; "*Valid*" representa el *valid bit*; "*Tag*" representa el valor decimal del tag que indica el bloque de memoria almacenado en la línea; y "*Time*" representa el tiempo **desde el último acceso a la línea**. A partir de este estado, se realiza el acceso a memoria de las siguientes direcciones: `0x29A`, `0x0FB`, `0x33C`, `0x7BD`, `0x7CE`. Indique, para cada acceso, si existe un *hit* o *miss* y, si corresponde, la línea de la caché que es modificada para la función de correspondencia *2-way associative*. Asuma una política de reemplazo LRU. Para responder, complete las tablas adjuntas al enunciado. El *tag* de cada línea lo puede escribir en base binaria, decimal o hexadecimal. No necesita indicar el *timestamp* final por línea, pero sí debe considerar su valor en caso de reemplazos.

Solución:

Al tener una memoria de 2048 palabras, se necesitan $\log_2(2048) = 11$ bits para cada dirección.Luego, al ser las líneas de 16 palabras, se requieren $\log_2(16) = 4$ bits de *offset* para ubicar una palabra dentro de una línea. Por otra parte, en una caché *2-way associative* se tienen conjuntos de 2 líneas, lo que deriva en un total de $\frac{16}{2} = 8$ conjuntos y, de esta forma, $\log_2(8) = 3$ bits de índice de conjunto. Esto, finalmente, resulta en $11 - 4 - 3 = 4$ bits de *tag*. Con esto en consideración, se obtiene el siguiente resultado de la secuencia de accesos a memoria:

1. Acceso a dirección `0x29A` $= 666 = 0101|001|1010b$. Resulta en **miss** al no existir línea válida con *tag* $0101b = 5$ en el conjunto 001 (líneas 2-3). Se copia el bloque sobre la línea 3 al estar desocupada. La línea 2 sigue siendo la que posee mayor *timestamp* dentro del conjunto, en caso de requerir un reemplazo.

2. Acceso a dirección `0x0FB` $= 251 = 0001|111|1011b$. Resulta en *miss* al no existir línea válida con *tag* $0001b = 1$ en el conjunto 111 (líneas 14-15). Se copia el bloque sobre la línea 15 al estar desocupada. La línea 14 sigue siendo la que posee mayor *timestamp* dentro del conjunto, en caso de requerir un reemplazo.

3. Acceso a dirección `0x33C` $= 828 = 0110|011|1100b$. Resulta en *hit*, ya que la línea 6

posee tag igual a $0110b = 6$ en el conjunto 011 (líneas 6-7). La línea 7 se convierte en la que posee mayor *timestamp* dentro del conjunto en caso de que se requiera un reemplazo.

4. Acceso a dirección `0x7BD` = 1981 = 1111|011|1101b. Resulta en *miss* al no existir línea válida con *tag* $1111b = 15$ en el conjunto 011 (líneas 6-7). Se copia el bloque sobre la línea 7, al ser esta la de mayor *timestamp*. Ahora, la línea 6 es la que posee mayor *timestamp* dentro del conjunto en caso de requerir un reemplazo.

5. Acceso a dirección `0x7CE` = 1998 = 1111|100|1110b. Resulta en *hit*, ya que la línea 8 posee *tag* igual a $1111b = 15$ en el conjunto 100 (líneas 8-9). La línea 9 sigue siendo la que posee mayor *timestamp* dentro del conjunto en caso de requerir un reemplazo.

A continuación, se muestra el estado final de la caché:

Línea	*Valid*	*Tag*
0	1	4
1	1	7
2	1	6
3	1	5
4	1	2
5	1	15
6	1	6
7	1	15

Línea	*Valid*	*Tag*
8	1	15
9	1	8
10	1	12
11	1	4
12	1	4
13	1	11
14	1	3
15	1	1

Se otorgan 0.8 ptos. por acceso correctamente descrito (i.e. obtención de *tag*; determinación de *hit* o *miss* y selección de línea a reemplazar). Si se reemplaza la línea incorrecta por *timestamp*, se otorgan 0.4 ptos. en el acceso correspondiente.

Excepcionalmente, se otorgan 3 ptos. del total si se realiza todo el procedimiento de forma correcta, pero con error de arrastre en el cálculo de bits de *offset*, índice de conjunto o *tag*; o bien si se parte contando del conjunto 1 y no del conjunto 0 al momento de seleccionar las líneas.

Pregunta 3: P1.C 2025-1

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	Validez	*Tag*	*Timestamp*
0	1	57	5
1	1	27	10
2	1	40	2
3	1	34	8

Línea	Validez	*Tag*	*Timestamp*
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`, `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: Al tener una memoria de 1024 palabras, se necesitan $\log_2(1024) = 10$ bits para cada dirección. Por otra parte, al ser las líneas de 16 palabras, se requiere $\log_2(16) = 4$ bits de *offset* para ubicar una palabra dentro de una línea. Por otra parte, al ser una caché *fully associative* se tendrán $10 - 4 = 6$ bits de *tag*. Dados estos valores, se puede observar que el dígito hexadecimal menos significativo de cada dirección será el *offset*; mientras que el resto será el *tag*, recordando que son 6 bits y no 8.

Con esto en consideración, se obtiene el siguiente resultado de la secuencia de accesos a memoria:

1. Acceso a dirección `0x3AA` $= 938 = 111010|1010b$. Resulta en *hit*, ya que la línea 4 posee *tag* igual a $111010b = 58 =$ `0x3A`. La línea 1 sigue siendo la que posee mayor *timestamp*, en caso de requerir un reemplazo.

2. Acceso a dirección `0x1B0` $= 432 = 011011|0000b$. Resulta en *hit*, ya que la línea 1 posee *tag* igual a $011011b = 27 =$ `0x1B`. Ahora, la línea 3 es la que posee mayor *timestamp*, en caso de requerir un reemplazo.

3. Acceso a dirección `0x00F` $= 15 = 000000|1111b$. Resulta en *hit*, ya que la línea 7 posee *tag* igual a $000000b = 0 =$ `0x00`. La línea 3 sigue siendo la que posee mayor *timestamp*, en caso de requerir un reemplazo.

4. Acceso a dirección `0x1FF` = 511 = 011111|1111b. Resulta en *miss* al no existir línea válida con *tag* 011111b = 31. Se copia el bloque sobre la línea 3, al ser esta la de mayor *timestamp*. Ahora, la línea 0 es la que posee mayor *timestamp*, en caso de requerir un reemplazo.

5. Acceso a dirección `0x01A` = 26 = 000001|1010b. Resulta en *miss* al no existir línea válida con *tag* 000001b = 1. Se copia el bloque sobre la línea 0, al ser esta la de mayor *timestamp*. Ahora, la línea 5 es la que posee mayor *timestamp*, en caso de requerir un reemplazo.

6. Acceso a dirección `0x2BB` = 699 = 101011|1011b. Resulta en *miss* al no existir línea válida con *tag* 101011b = 43. Se copia el bloque sobre la línea 5, al ser esta la de mayor *timestamp*. Ahora, la línea 6 es la que posee mayor *timestamp*, en caso de requerir un reemplazo.

A continuación, se muestra el estado final de la caché:

Línea	Valid	Tag
0	1	1
1	1	27
2	1	40
3	1	31

Línea	Valid	Tag
4	1	58
5	1	43
6	1	62
7	1	0

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

Se asignan 0.6 ptos. por obtener el *hit-rate* correcto (ya sea explicitado u obtenido correctamente por cantidad de *hits* y *misses*) y 0.4 ptos por acceso correctamente descrito (i.e. obtención de tag; determinación de *hit* o *miss* y selección de línea a reemplazar). Si se reemplaza la línea incorrecta por timestamp, se otorgan 0.2 ptos. en el acceso correspondiente. Si se realiza todo de forma correcta, pero tomando un tag de 8 bits y no de 6, se descuentan 0.5 ptos. del total.

Feedback ayudantía

Escanee el QR para entregar *feedback* sobre la ayudantía.

