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

IIC2343 - Arquitectura de Computadores (II/2024)

Interrogación 3

Pauta de evaluación

Pregunta 1: Jerarquía de memoria (6 ptos.)

(a) (1 pto.) ¿Qué incidencia tiene la política de reemplazo en una caché con función de correspondencia *directly mapped*? Justifique.

> **Solución:** No tiene ninguna incidencia. En la función de correspondencia *directly mapped*, los reemplazos solo se realizan sobre una línea que ya se encuentra ocupada contenido con un bloque de memoria distinto al que se busca acceder. Por este motivo, no se utiliza ningún criterio de elección de línea a ser reeemplazada. Se otorgan **0.5 ptos.** por responder correctamente; y **0.5 ptos.** por justificación.

(b) (1 pto.) ¿Cómo incide el parámetro "N" en el *hit-time* de una memoria caché *N-way associative*?

> **Solución:** El parámetro N corresponde a la cantidad de líneas por conjunto en una caché *N-way associative*. Luego, el *hit-time* en este tipo de caché corresponde al tiempo de búsqueda de un bloque en las líneas del conjunto al que se mapea. Por ende, a mayor N, mayor será el *hit-time* al tener que revisar una mayor cantidad de líneas en el conjunto. Se otorgan **0.5 ptos.** por señalar correctamente la incidencia; y **0.5 ptos.** por la explicación.

(c) (4 ptos.) Suponga que, para una memoria principal de 4096 palabras de 1 byte, posee una caché de 32 líneas y 16 palabras por línea con el siguiente estado intermedio:

Línea	*Valid*	*Tag*	*Time*
0	1	0	7
1	1	7	11
2	1	10	17
3	0	12	-
4	1	12	15
5	1	15	4
6	0	1	-
7	0	8	-

Línea	*Valid*	*Tag*	*Time*
8	1	5	13
9	1	9	5
10	1	13	9
11	1	11	2
12	1	15	10
13	1	0	12
14	1	1	21
15	1	5	14

Línea	*Valid*	*Tag*	*Time*
16	0	10	-
17	0	15	-
18	0	2	-
19	0	7	-
20	0	9	-
21	0	14	-
22	0	1	-
23	0	2	-

Línea	*Valid*	*Tag*	*Time*
24	1	8	6
25	1	9	16
26	1	14	18
27	1	7	8
28	1	3	3
29	1	6	19
30	1	2	20
31	1	15	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: 0xC27, 0xFF0, 0x123, 0x782. 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 la función de correspondencia *2-way associative*. Asuma una la 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 considerar su valor en caso de reemplazos.

Solución: Al tener una memoria de 4096 palabras, se necesitan $\log_2(4096) = 12$ 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, en una caché *2-way associative* se tienen conjuntos de 2 líneas, lo que deriva en un total de $\frac{32}{2} = 16$ conjuntos y, de esta forma, $\log_2(16) = 4$ bits de índice de conjunto. Esto, finalmente, resulta en $12 - 4 - 4 = 4$ bits de *tag*. Dados estos valores, se puede observar que el dígito hexadecimal menos significativo de cada dirección será el *offset*; el dígito del medio será el identificador de conjunto; y el dígito más significativo será el *tag*.

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

1. Acceso a dirección 0xC27 = 3111 = 1100|0010|0111b. Resulta en *hit*, ya que la línea 4 posee *tag* igual a 1100b = 12 = 0xC en el conjunto 0x2 = 2 (líneas 4 y 5). Ahora, la línea 5 es la que posee mayor *timestamp*, en caso de requerir un reemplazo.
2. Acceso a dirección 0xFF0 = 4080 = 1111|1111|0000b. Resulta en *hit*, ya que la línea 31 posee *tag* igual a 1111b = 15 = 0xF en el conjunto 0xF = 15 (líneas 30 y 31). Ahora, la línea 30 es la que posee mayor *timestamp*, en caso de requerir un reemplazo.
3. Acceso a dirección 0x123 = 291 = 0001|0010|0011b. Resulta en *miss* al no existir línea válida con *tag* 0001b = 1 en el conjunto 0x2 = 2 (líneas 4 y 5). Se copia el bloque sobre la línea 5, al ser esta la de mayor *timestamp*. Ahora, la línea 4 es la que posee mayor *timestamp*, en caso de requerir un reemplazo.
4. Acceso a dirección 0x782 = 1922 = 0111|1000|0010b. Resulta en *miss* al no existir línea válida con *tag* 0111b = 7 en el conjunto 0x8 = 8 (líneas 16 y 17). Se copia el bloque sobre la línea 16, al ser esta la primera disponible.

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

Línea	*Valid*	*Tag*
0	1	0
1	1	7
2	1	10
3	0	12
4	1	12
5	1	1
6	0	1
7	0	8

Línea	*Valid*	*Tag*
8	1	5
9	1	9
10	1	13
11	1	11
12	1	15
13	1	0
14	1	1
15	1	5

Línea	*Valid*	*Tag*
16	1	7
17	0	15
18	0	2
19	0	7
20	0	9
21	0	14
22	0	1
23	0	2

Línea	*Valid*	*Tag*
24	1	8
25	1	9
26	1	14
27	1	7
28	1	3
29	1	6
30	1	2
31	1	15

Hit-rate: $\frac{2}{4} = 0{,}5 = 50\,\%$

Se asignan **0.4 ptos.** por obtener el *hit-rate* correcto (ya sea explicitado u obtenido correctamente por cantidad de *hits* y *misses*) y **0.9 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 **0.5 ptos.** en el acceso correspondiente.

Excepcionalmente, se otorgan **3 ptos.** del total si se realiza **todo** el procedimiento y cálculo de *hit-rate* 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 2: Multiprogramación (6 ptos.)

(a) (2 ptos.) Considere la siguiente sentencia: "Si en un esquema de paginación el tamaño de página se duplica, entonces el tamaño de la tabla de páginas se reduce **exactamente** a la mitad". ¿Es esta sentencia correcta? Justifique.

> **Solución:** La sentencia es **incorrecta**. Al duplicar el tamaño de entradas, la cantidad de páginas disminuye a la mitad. Esto deriva en que la cantidad de entradas de la tabla de páginas también se reduzca a la mitad. No obstante, la cantidad de marcos físicos también se reduce a la mitad. Esto implica un bit menos de número de marco físico y, por ende, un bit menos en cada entrada de la tabla de páginas. Por lo tanto, el tamaño de la tabla de páginas se reduce a **menos de la mitad**. Se otorga **1 pto.** por señalar que la sentencia es incorrecta; y **1 pto.** por justificar correctamente.
>
> Si solo se señala que la sentencia es incorrecta sin ningún tipo de justificación, **no se otorga puntaje**.
>
> Si se señala que la sentencia es correcta, pero se **explicita** que la cantidad de entradas de la tabla de páginas disminuye a la mitad, se otorga **1 pto.**

(b) (4 ptos.) Suponga que posee un espacio de direcciones virtuales de 16 bits, direcciones físicas de 16 bits y un tamaño de páginas de 4KiB. En este esquema, un proceso en ejecución posee el siguiente estado para su tabla de páginas en un instante dado:

Entrada	Presente	Marco
0	0	15
1	0	13
2	0	5
3	0	7
4	0	9
5	0	1
6	1	0
7	1	4

Entrada	Presente	Marco
8	1	2
9	1	14
10	1	1
11	1	13
12	1	5
13	1	15
14	1	11
15	1	12

La columna "Entrada" representa el número de entrada de la tabla; la columna "Presente" representa si el contenido del marco físico se encuentra en la memoria principal (1) o en el *swap file* (0); y "Marco" representa el valor, en base decimal, del marco físico asociado a la página ligada a la entrada. Con este estado de tabla de páginas, se realiza el acceso a memoria a las siguientes direcciones virtuales: 0xBEAD, 0xD00D, 0xF001, 0xCABE. 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 las direcciones físicas obtenidas en base binaria, decimal o hexadecimal.

Solución: Al tener un tamaño de páginas de 4KiB, se tiene un total de $2^2 \times 2^{10} = 2^{12}$ palabras de memoria por página, por lo que se tienen $\log_2(2^{12}) = 12$ bits de *offset*. De esto se deduce que el número de marco físico es de $16 - 12 = 4$ bits y el número de página es de $16 - 12 = 4$ bits. Este último coincide con las dimensiones de la tabla de páginas, dado que se tienen $2^4 = 16$ entradas. Al tener 4 bits de número de página, el dígito hexadecimal más significativo de cada dirección virtual nos entrega directamente el número de página. Más aún, como también se tienen 4 bits de número de marco físico, se puede reemplazar directamente el dígito hexadecimal del número de página por el valor hexadecimal obtenido de la tabla. Así, se tiene el siguiente resultado de la secuencia de traducciones:

1. Dirección 0xBEAD $\rightarrow$ página 0xB = 11. Se tiene un marco físico **presente en memoria** con valor igual a 13 = 1101b = 0xD. Dirección física traducida = 0xDEAD.
2. Dirección 0xD00D $\rightarrow$ página 0xD = 13. Se tiene un marco físico **presente en memoria** con valor igual a 15 = 1111b = 0xF. Dirección física traducida = 0xF00D.
3. Dirección 0xF001 $\rightarrow$ página 0xF = 15. Se tiene un marco físico **presente en memoria** con valor igual a 12 = 1100b = 0xC. Dirección física traducida = 0xC001.
4. Dirección 0xCABE $\rightarrow$ página 0xC = 12. Se tiene un marco físico **presente en memoria** con valor igual a 5 = 0101b = 0x5. Dirección física traducida = 0x5ABE.

Se asigna **1 pto.** 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 otorgan como máximo **0.5 ptos.** por traducción 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: Paralelismo a nivel de instrucción (ILP) (6 ptos.)

(a) (2 ptos.) La arquitectura del computador básico con *pipeline* no permite la ejecución de la instrucción de salto `JGE`. Explique, a partir del diagrama adjunto, por qué no es posible su ejecución y describa qué modificaciones habría que realizar para implementarla. No es necesario que modifique el diagrama ni que dibuje circuitos, solo que describa los cambios necesarios. Puede asumir que la ALU cuenta con todos los circuitos internos del computador básico sin *pipeline*.

Solución: No se puede ejecutar la instrucción `JGE` por la **falta de la *flag* `N`**. El salto `JGE` se realiza si, y solo si $A \geq B$, lo que se evalúa con la *flag* `N` igual a 0. Para su ejecución, asumiendo que se cuenta con todos los circuitos internos de la ALU del computador básico sin *pipeline*, se pueden aplicar las siguientes modificaciones:

- Conectar la salida de la *flag* `N` de la ALU con el registro intermedio `EX/MEM`.
- Transmitir la señal `N` a la *Jump Unit*.
- Modificar la *Jump Unit* para que evalúe la condición `N` igual a 0 en caso de ejecutar una instrucción `JGE`.

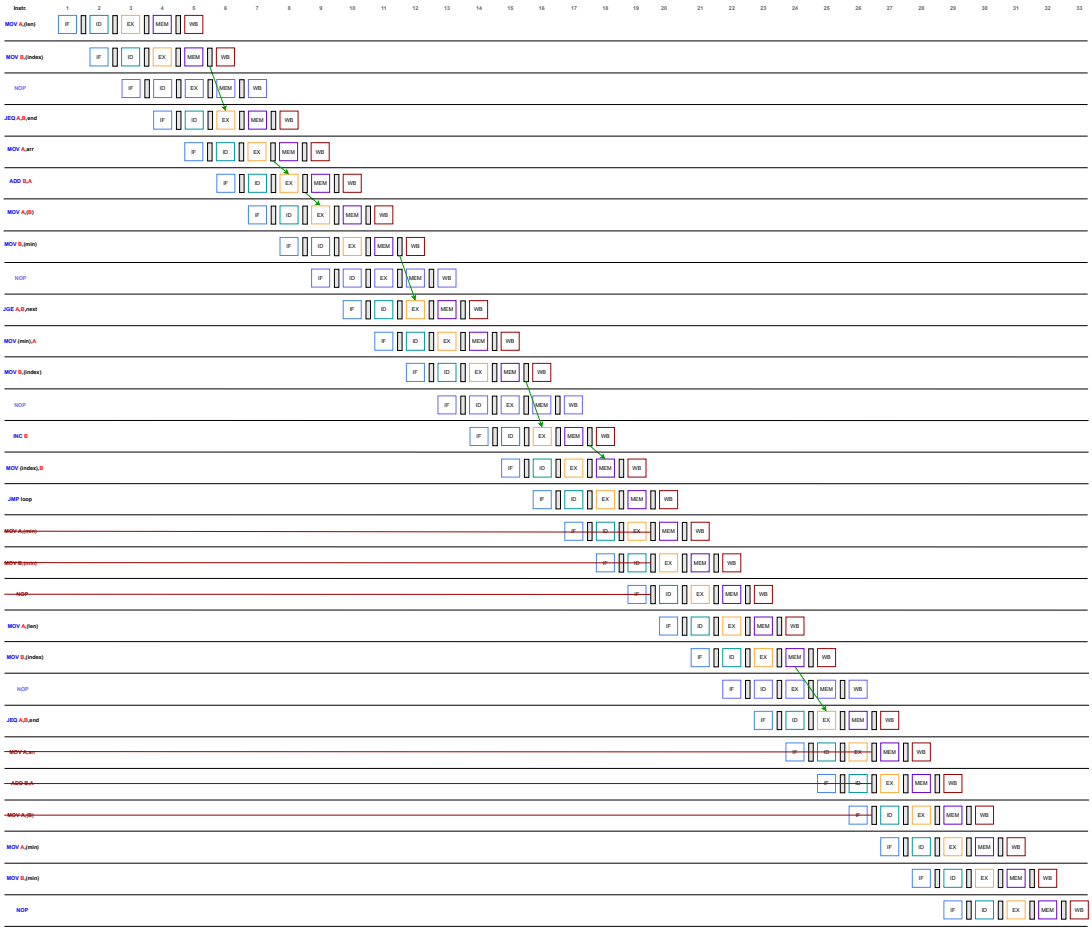
Se otorgan **0.5 ptos.** por explicar por qué no se puede ejecutar la instrucción `JGE`; **0.5 ptos.** por señalar que se debe evaluar la condición `N` igual a cero (ya sea como parte de la explicación o la modificación); y **1 pto.** por describir una modificación que permita hacer uso de la *flag* `N` en la *Jump Unit*, independiente de si se transmite de la ALU o si se computa de otra forma. Para la modificación, se otorga la mitad del puntaje si existe, como máximo, **un error** en la descripción entregada.

(b) (4 ptos.) A partir del siguiente programa del computador básico con *pipeline* y la instrucción `JGE` implementada:

```
DATA:
  len 4
  arr 10
      3
      8
      1
  min 3
  index 3
CODE:
  loop:
    MOV A, (len)
    MOV B, (index)
    JEQ A, B, end
    MOV A, arr
    ADD B, A
    MOV A, (B)
    MOV B, (min)
    JGE A, B, next
    update_min:
      MOV (min), A
    next:
      MOV B, (index)
      INC B
      MOV (index), B
      JMP loop
    end:
      MOV A,(min)
      MOV B,(min)
      NOP
```

Determine el número de ciclos que demora su ejecución detallando en un diagrama los estados del *pipeline* por instrucción. Asuma que el manejo de *stalling* es por *software* a través de la instrucción `NOP` y que la unidad predictora de saltos asume que estos nunca se realizan, independiente de que estos sean condicionales o incondicionales. Indique en el diagrama adjunto al enunciado cuando ocurre *forwarding* (con flechas desde el registro hacia la etapa que corresponda), *stalling* y *flushing* (con tachado en las instrucciones *flusheadas*).

Solución: A continuación, se muestra el *pipeline* resultante de la ejecución del programa.



Del *pipeline* anterior se deduce que el programa termina de correr después de 33 ciclos. No obstante, también se considera correcto que se señale un valor igual a 32 si no se toma en cuenta la instrucción `NOP` del final. Se otorga puntaje de la siguiente forma: **0.25 ptos.** por cada *stalling* correctamente detectado (**1 pto.** por todas las detecciones); **0.5 ptos.** por cada *flushing* correctamente detectado (**1 pto.** por todas las detecciones); **0.25 ptos.** por cada *forwarding* correctamente detectado (**1.75 ptos.** por todas las detecciones); y **0.25 ptos.** por la deducción correcta de la cantidad de ciclos en la que corre el programa.

Si no se agrega correctamente un *stall*, pero se aplican bien los *forwardings*, se otorga puntaje en dicho criterio. No obstante, si existen uno o más *forwardings* mal aplicados, se descuentan **0.25 ptos.**

Se otorga el puntaje por cantidad de ciclos solo si se obtiene el valor correcto por la ejecución correcta del programa y **no por accidente por errores de ejecución**.

No hay descuento si se incluyen las instrucciones de escritura de memoria correspondientes al segmento `DATA`, pero solo si la ejecución de estas se describe de forma correcta.

Pregunta 4: Coherencia de Caché (6 ptos.)

Suponga que posee una arquitectura MIMD de memoria compartida con 3 CPU que poseen su propia caché y que siguen el protocolo MESI para asegurar consistencia. La memoria contiene los siguientes datos de variables; mientras que CPU0, CPU1 y CPU2 ejecutan los siguientes programas:

Dirección	*Label*	Valor
0x00	`var1`	255
0x01	`var2`	253
...	...	...
0xFE		0
0xFF		0

```
// CPU0                 // CPU1                 // CPU2
MOV A, (var1)           MOV A, 0             MOV B, 0
MOV B, (var2)           NOP                  PUSH B
AND B,A                 PUSH A               INC B
INC B                   POP B                PUSH B
MOV B, (B)                                   POP A
MOV (B), B              MOV B,(B)
```

Asumiendo que cada dirección se almacena en una línea distinta, y que el contador `SP` posee un valor inicial igual a 255 en **todas** las CPU, indique el estado de cada línea de las caché (`M`/`E`/`S`/`I`) para cada ciclo completando la tabla adjunta, asumiendo que todas las líneas parten en estado `I` (ciclo 0):

	CPU0				**CPU1**				**CPU2**			
Ciclo	**0x00**	**0x01**	**0xFE**	**0xFF**	**0x00**	**0x01**	**0xFE**	**0xFF**	**0x00**	**0x01**	**0xFE**	**0xFF**
0	I	I	I	I	I	I	I	I	I	I	I	I
1												
2												
3												
4												
5												
6												

Solución: A continuación se muestra la tabla final esperada por cada iteración:

	CPU0				CPU1				CPU2			
Ciclo	**0x00**	**0x01**	**0xFE**	**0xFF**	**0x00**	**0x01**	**0xFE**	**0xFF**	**0x00**	**0x01**	**0xFE**	**0xFF**
0	I	I	I	I	I	I	I	I	I	I	I	I
1	E	I	I	I	I	I	I	I	I	I	I	I
2	E	E	I	I	I	I	I	I	I	I	I	M
3	E	E	I	I	I	I	I	M	I	I	I	I
4	E	E	I	I	I	I	I	M	I	I	M	I
5	E	E	S	I	I	I	I	M	I	I	S	I
6	S	M	S	I	S	I	I	M	I	I	S	I

Para llegar al estado final correcto, es importante haber tomado en cuenta lo siguiente:

- Si se realiza la operación `AND B,A` con 8 bits para cada registro, se llega al resultado `B = 253`, por lo que la CPU0 luego accede a la dirección de memoria 254 = 0xFE considerando la ejecución de la instrucción `INC B`.
- La ejecución de `POP A` y `POP B` toma dos ciclos. La lectura del valor de memoria se realiza en el **segundo ciclo**, dado que el primero lo único que hace es incrementar el valor de `SP`.
- Al ser `SP` parte de una CPU, su valor es **independiente** de lo que ejecuten otras. Es decir, que una CPU realice un `PUSH` o un `POP` no afecta el valor de `SP` de otra.

Se otorga **1 pto.** por cada iteración (fila) con estado final correcto para todas las CPU. Se otorgan **0.7 ptos.** por iteración (fila) si se tiene como máximo una celda con estado erróneo; y **0.3 ptos.** si se tienen como máximo dos celdas con estado erróneo.

Para efectos de corrección, toda celda vacía se considerará en estado inválido, a excepción del caso en el que no se haya contestado la pregunta.

Si una celda se modifica a un estado erróneo en una iteración y se mantiene en la siguiente, solo se descuenta en la fila donde ocurrió el error. En cambio, si debía cambiar su estado en la siguiente iteración y se mantiene en el estado erróneo, se descuenta en ambas filas.

Por último, si un error de interpretación de código genera un estado erróneo en múltiples celdas, se aplica el descuento solo sobre una de ellas si los cambios de todas las celdas son coherentes con la interpretación.