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

IIC2343 - Arquitectura de Computadores (II/2022)

Interrogación 2

Respuestas sin desarrollo o justificación no tendrán puntaje.

Jueves 27 de Octubre a las 18:30 horas

Instrucciones

Lea atentamente los enunciados. Responda cada pregunta en hojas separadas. Ponga su nombre, número de alumno y número de lista. Siga el código de honor. Existen 4 preguntas de las cuales deberá contestar un máximo de 3 preguntas.

Código de Honor de la UC

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

Pregunta	Puntos	Logrados
Caché	20	
Memoria Virtual	20	
ILP (*Instruction Level Parallelism*)	20	
IO y RISC-V	20	
Total:	80	

Nombre: _____ **N° de alumno**: _____ **N° lista**: ____

Pregunta 1: Caché (20 ptos.)

Considera un computador con una memoria de 1 KiB, organizada en palabras de 4 bytes cada una. Las direcciones generadas por el programa son siempre direcciones de palabras, es decir, la dirección del byte (de la palabra) con la dirección numéricamente menor.

Para este computador se está diseñando el sistema de caché, con líneas de cuatro palabras, y se quiere decidir si es mejor un sistema 2-way set associative o uno 4-way set associative, considerando que en la caché caben 8 líneas en total y con política de escritura *write-through*.

(a) ¿Cuántos bits adicionales—para tags, validez, y offset—necesita cada una de las dos versiones del sistema de caché? Justifica. [No consideres la implementación del sistema de reemplazo de líneas.] (5)

Solución:

- **2-way set associative:**
  - Se requiere un bit de validez.
  - Como se tienen lineas de 4 palabras, se requieren 2 bits para el offset.
  - Como se tiene una cache de 8 lineas, ordenada en sets de 2 lineas cada uno, tenemos 4 sets. Se necesitan 2 bits para el indicar el set, por lo que se necesitan 4 para el tag.

  La descomposición de una dirección se vería como:

$$\underbrace{0001}_{tag}\ \underbrace{10}_{indice}\ \underbrace{01}_{offset}\ 00$$

- **4-way set associative:**
  - Se requiere un bit de validez.
  - Como se tienen lineas de 4 palabras, se requieren 2 bits para el offset.
  - Como se tiene una cache de 8 lineas, ordenada en sets de 4 lineas cada uno, tenemos 2 sets. Se necesita 1 bit para el indicar el set, por lo que se necesitan 5 para el tag.

  La descomposición de una dirección se vería como:

$$\underbrace{00011}_{tag}\ \underbrace{0}_{indice}\ \underbrace{01}_{offset}\ 00$$

Considera la siguiente secuencia de direcciones de memoria generadas como consecuencia de la ejecución de un programa:

1. 0x028 (40)
2. 0x048 (72)
3. 0x014 (20)
4. 0x078 (120)
5. 0x010 (16)
6. 0x070 (112)
7. 0x084 (132)
8. 0x008 (8)
9. 0x064 (100)
10. 0x058 (88)
11. 0x0B4 (180)
12. 0x024 (36)
13. 0x044 (68)
14. 0x098 (152)
15. 0x060 (96)
16. 0x094 (148)
17. 0x018 (24)
18. 0x004 (4)

(b) Suponiendo que la estrategia de reemplazo de líneas fuera la ideal (Bélády), es decir, que el computador contara con un dispositivo que pudiera determinar cuál de las líneas en juego en la caché es la que se va a demorar más en volver a ser usada en el futuro, ¿cuál es el *hit rate* para cada una de las dos versiones del sistema de caché? (15)

Solución:

2-way set associative

Acceso (Decimal)	Acceso (Binario)	Indice	Tag	Set 0		Set 1		Set 2		Set 3		Hit	Comentario
				Tag 0	Tag 1	Tag 0	Tag 1	Tag 0	Tag 1	Tag 0	Tag 1		
40	0000101000	10	0000	-	-	-	-	0000	-	-	-	0	Como tiene indice 10, se debe guardar en el set 2.
72	0001001000	00	0001	0001	-	-	-	0000	-	-	-	0	Como tiene indice 00, se debe guardar en el set 0.
20	0000010100	01	0000	0001	-	0000	-	0000	-	-	-	0	Como tiene indice 01, se debe guardar en el set 1.
120	0001111000	11	0001	0001	-	0000	-	0000	-	0001	-	0	Como tiene indice 11, se debe guardar en el set 3.
16	0000010000	01	0000	0001	-	0000	-	0000	-	0001	-	1	Hit!
112	0001110000	11	0001	0001	-	0000	-	0000	-	0001	-	1	Hit!
132	0010000100	00	0010	0001	0010	0000	-	0000	-	0001	-	0	Como tiene indice 00, se debe guardar en el set 0. Toma el segundo espacio.
8	0000001000	00	0000	0001	0000	0000	-	0000	-	0001	-	0	Se reemplaza la linea con tag 0010 porque no se vuelve a usar en el futuro.
100	0001100100	10	0001	0001	0000	0000	-	0000	0001	0001	-	0	Como tiene indice 10, se debe guardar en el set 2. Toma el segundo espacio.
88	0001011000	01	0001	0001	0000	0000	0001	0000	0001	0001	-	0	Como tiene indice 01, se debe guardar en el set 1. Toma el segundo espacio.
180	0010110100	11	0010	0001	0000	0000	0001	0000	0001	0001	0010	0	Como tiene indice 11, se debe guardar en el set 3. Toma el segundo espacio.
36	0000100100	10	0000	0001	0000	0000	0001	0000	0001	0001	0010	1	Hit!
68	0001000100	00	0001	0001	0000	0000	0001	0000	0001	0001	0010	1	Hit!
152	0010011000	01	0010	0001	0000	0000	0010	0000	0001	0001	0010	0	Se reemplaza la linea con tag 0001 porque no se vuelve a usar en el futuro.
96	0001100000	10	0001	0001	0000	0000	0010	0000	0001	0001	0010	1	Hit!
148	0010010100	01	0010	0001	0000	0000	0010	0000	0001	0001	0010	1	Hit!
24	0000011000	01	0000	0001	0000	0000	0010	0000	0001	0001	0010	1	Hit!
4	0000000100	00	0000	0001	0000	0000	0010	0000	0001	0001	0010	1	Hit!

El *hit-rate* es de $\frac{8}{18}$.

4-way set associative

Acceso (Decimal)	Acceso (Binario	Indice	Tag	Set 0				Set 1				Hit	Comentario
				Tag 0	Tag 1	Tag 2	Tag 3	Tag 0	Tag 1	Tag 2	Tag 3		
40	0000101000	0	00001	00001	-	-	-	-	-	-	-	0	Como tiene indice 0, se debe guardar en el set 0.
72	0001001000	0	00010	00001	00010	-	-	-	-	-	-	0	Como tiene indice 0, se debe guardar en el set 0. En la segunda posicion.
20	0000010100	1	00000	00001	00010	-	-	00000	-	-	-	0	Como tiene indice 1, se debe guardar en el set 1.
120	0001111000	1	00011	00001	00010	-	-	00000	00011	-	-	0	Como tiene indice 1, se debe guardar en el set 1. En la segunda posicion.
16	0000010000	1	00000	00001	00010	-	-	00000	00011	-	-	1	Hit!
112	0001110000	1	00011	00001	00010	-	-	00000	00011	-	-	1	Hit!
132	0010000100	0	00100	00001	00010	00100	-	00000	00011	-	-	0	Como tiene indice 0, se debe guardar en el set 0. En la tercera posicion.
8	0000001000	0	00000	00001	00010	00100	00000	00000	00011	-	-	0	Como tiene indice 0, se debe guardar en el set 0. En la cuarta posicion.
100	0001100100	0	00011	00001	00010	00011	00000	00000	00011	-	-	0	Se reemplaza la linea con tag 00100 porque no se vuelve a usar en el futuro.
88	0001011000	1	00010	00001	00010	00011	00000	00000	00011	00010	-	0	Como tiene indice 1, se debe guardar en el set 1. En la tercera posicion.
180	0010110100	1	00101	00001	00010	00011	00000	00000	00011	00010	00101	0	Como tiene indice 1, se debe guardar en el set 1. En la cuarta posicion.
36	0000100100	0	00001	00001	00010	00011	00000	00000	00011	00010	00101	1	Hit!
68	0001000100	0	00010	00001	00010	00011	00000	00000	00011	00010	00101	1	Hit!
152	0010011000	1	00100	00001	00010	00011	00000	00000	00100	00010	00101	0	Se reemplaza la linea con tag 00011 porque no se vuelve a usar en el futuro. Tambien se pudo haber cambiado la con tag 00010 o 00101
96	0001100000	0	00011	00001	00010	00011	00000	00000	00100	00010	00101	1	Hit!
148	0010010100	1	00100	00001	00010	00011	00000	00000	00100	00010	00101	1	Hit!
24	0000011000	1	00000	00001	00010	00011	00000	00000	00100	00010	00101	1	Hit!
4	0000000100	0	00000	00001	00010	00011	00000	00000	00100	00010	00101	1	Hit

El *hit-rate* es de $\frac{8}{18}$.

Pregunta 2: Memoria Virtual (20 ptos.)

Se tiene un computador con memoria RAM de 16 KiB, sin embargo, los programas tienen un espacio direccionable virtual de 64 KiB. Además, se sabe que los *page frames*/marcos son de 4 KiB.

Por otra parte, ustedes saben que su computador emplea tablas de paginas en donde cada entrada almacena los *bits* suficientes para indicar el marco y todos los bits de metadata necesarios para el funcionamiento y criterio de reemplazo, el cual sera LRU. Pueden suponer que las tablas de paginas no ocupan espacio en la memoria.

Finalmente, las especificaciones de su computador indican que este cuenta con un con una unidad TLB, capaz de almacenar tan solo 2 traducciones. El criterio de reemplazo para esta es FIFO.

Considere que tanto la tabla de paginas y la TLB comienzan vacías. En base, a la información anterior y la siguiente secuencia de accesos a memoria (direcciones virtuales) indique:

Accesos a memoria:

1. 0xE367
2. 0xA306
3. 0xE492
4. 0x5F7C
5. 0x737F
6. 0x3D64
7. 0x0A64
8. 0x136E
9. 0xEC17
10. 0x0767
11. 0xE648
12. 0xA000
13. 0x587D
14. 0x03BF
15. 0xDDBB
16. 0x6450

(a) Cantidad de *page frames* de la memoria física (2)

Solución:

Para calcular la cantidad de marcos que tiene la memoria física se debe dividir la capacidad de la memoria física por la capacidad de un marco.

$$\frac{2^4 \cdot 2^{10}}{2^2 \cdot 2^{10}} = 2^2 = 4 \text{ marcos}$$

(b) Cantidad de paginas de la memoria virtual (2)

Solución:

Para calcular la cantidad de paginas que tiene la memoria virtual se debe dividir la capacidad de la memoria virtual por la capacidad de una pagina.

$$\frac{2^6 \cdot 2^{10}}{2^2 \cdot 2^{10}} = 2^4 = 16 \text{ paginas}$$

(c) Cantidad de *page faults* (8)

Solución:

Acceso	Pagina	Marco 0		Marco 1		Marco 2		Marco 3		Page Fault	Comentario
		Pagina	LRU	Pagina	LRU	Pagina	LRU	Pagina	LRU		
0xE367	E	E	1	-	-	-	-	-	-	1	Se puede asignar el marco 0 a la pagina E.
0xA306	A	E	2	A	1	-	-	-	-	1	Se puede asignar el marco 1 a la pagina A.
0XE492	E	E	1	A	2	-	-	-	-	0	Hit!
0x5F7C	5	E	2	A	3	5	1	-	-	1	Se puede asignar el marco 2 a la pagina 5.
0x737F	7	E	3	A	4	5	2	7	1	1	Se puede asignar el marco 3 a la pagina 7.
0x3D64	3	E	4	3	1	5	3	7	2	1	Dado que todos los marcos están tomados, se reemplaza por el que lleva mas tiempo sin usar. En este caso es el marco 1.
0x0A64	0	0	1	3	2	5	4	7	3	1	Dado que todos los marcos están tomados, se reemplaza por el que lleva mas tiempo sin usar. En este caso es el marco 0.
0x136E	1	0	2	3	3	1	1	7	4	1	Dado que todos los marcos están tomados, se reemplaza por el que lleva mas tiempo sin usar. En este caso es el marco 2.
0xEC17	E	0	3	3	4	1	2	E	1	1	Dado que todos los marcos están tomados, se reemplaza por el que lleva mas tiempo sin usar. En este caso es el marco 3.
0x0767	0	0	1	3	5	1	3	E	2	0	Hit!
0xE648	E	0	2	3	6	1	4	E	1	0	Hit!
0xA000	A	0	3	A	1	1	5	E	2	1	Dado que todos los marcos están tomados, se reemplaza por el que lleva mas tiempo sin usar. En este caso es el marco 1.
0x587D	5	0	4	A	2	5	1	E	3	1	Dado que todos los marcos están tomados, se reemplaza por el que lleva mas tiempo sin usar. En este caso es el marco 2.
0x03BF	0	0	1	A	3	5	2	E	4	0	Hit!
0xDDBB	D	0	2	A	4	5	3	D	1	1	Dado que todos los marcos están tomados, se reemplaza por el que lleva mas tiempo sin usar. En este caso es el marco 3.
0x6450	6	0	3	6	1	5	4	D	2	1	Dado que todos los marcos están tomados, se reemplaza por el que lleva mas tiempo sin usar. En este caso es el marco 1.

La cantidad de *page faults* es 12.

(d) *Hit-rate* de la TLB (8)

Solución:

Acceso	Pagina	TLB 0		TLB 1		Hit	Comentario
		Pagina	FIFO	Pagina	FIFO		
0xE367	E	E	1	-	-	0	
0xA306	A	E	1	A	2	0	
0XE492	E	E	1	A	2	1	Hit!
0x5F7C	5	5	3	A	2	0	
0x737F	7	5	3	7	4	0	
0x3D64	3	3	5	7	4	0	
0x0A64	0	3	5	0	6	0	
0x136E	1	1	7	0	6	0	
0xEC17	E	1	7	E	8	0	
0x0767	0	0	9	E	8	0	
0xE648	E	0	9	E	8	1	Hit!
0xA000	A	0	9	A	10	0	
0x587D	5	5	11	A	10	0	
0x03BF	0	5	11	0	12	0	
0xDDBB	D	D	13	0	12	0	
0x6450	6	D	13	6	14	0	

El *hit-rate* es $\frac{2}{16}$.

Pregunta 3: ILP (*Instruction Level Parallelism*) (20 ptos.)

Considerando un computador básico con *pipeline* que tiene: 5 etapas, *forwarding* entre todas sus etapas, manejo de *stalling* por software (instrucción NOP) y predicción de salto asumiendo que no ocurre. Y la ejecución efectiva del siguiente código:

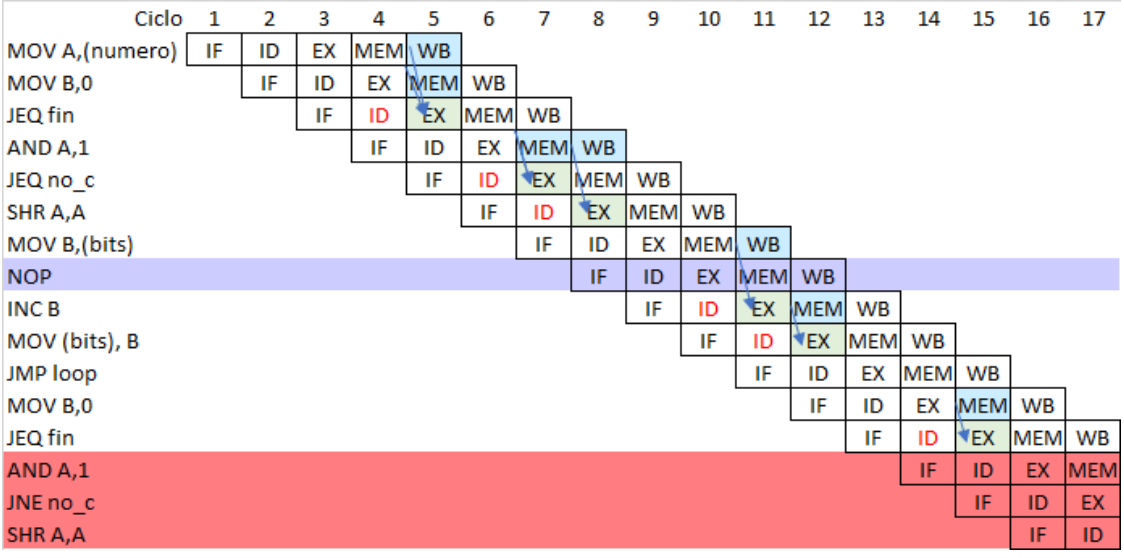
```
DATA:
      numero 1
      bits 0

CODE:

      MOV A,(numero)
loop: MOV B,0
      JEQ fin
      AND A,1
      JEQ no_c
      SHR A,A
      MOV B,(bits)
      INC B
      MOV (bits),B
no_c: JMP loop
fin:
```

(a) Realice un diagrama los estados del *pipeline* por instrucción. Indique en el diagrama cuándo ocurre *forwarding*, *stalling* y *fushing*. Asuma que data ya se encuentra en la memoria de datos. (15)

Solución:



Flechas indican *forwarding*, instrucciones en morado *stalling* y en rojo *flushing*

(b) Indique cómo podría reordenar el código para reducir la cantidad de ciclos que toma su ejecución. (5)

Solución:

```
DATA:
      numero 1
      bits 0

CODE:

      MOV A,(numero)
loop: MOV B,0
      JEQ fin
      AND A,1
      JEQ no_c
      MOV B,(bits)
      SHR A,A // Con esta instrucción aquí ya no es necesaria la burbuja
      INC B
      MOV (bits),B
no_c: JMP loop
fin:
```

Si movemos una instrucción donde actualmente tenemos que hacer *stalling*, entonces ya no sería necesario agregar un NOP. La instrucción que podemos mover sin afectar el resultado del programa es SHR A,A. Esto nos reduce la ejecución en 1 ciclo.

Pregunta 4: IO y RISC-V (20 ptos.)

Una empresa de instrumentalización ambiental le ha solicitado realizar la interfaz análogo-digital para un pluviómetro.

El elemento electromecánico del pluviometro es un botón que es presionado brevemente por un balancín que alterna al llenarse de agua en uno de sus extremos. El volumen de llenado es conocido, por lo que al contar la cantidad de veces que es presionado el botón podemos saber la cantidad de agua caída.

Como componente de control se le entrega un microcontrolador FU540-C000 de SiFive, que implementa la ISA de RISC-V. Dicho microcontrolador tiene puertos de entrada y salida digitales llamados GPIO (General Purpose Input/Ouput), cuyos registros de estado y control son de 32 bits y se encuentran mapeados en memoria desde la dirección 0x10060000, son los siguientes:

Offset	Name	Description
0x00	input_val	Contiene el valor de entrada
0x04	input_en	Controla si está habilitado como una entrada
0x08	output_en	Controla si está habilitado como una salida
0x0C	output_val	Controla el valor de salida
0x18	rise_ie	Controla si está habilitada las interrupciones por flanco de subida
0x20	fall_ie	Controla si está habilitada las interrupciones por flanco de bajada

Cada uno de los 32 bits de estos registros controla independientemente una de de 32 señales digitales de entrada o salida. De las cuales en este microcontrolador tiene disponible solo las 16 menos significativas, GPIOs del 0 al 15.

Su objetivo será detectar cuándo se acciona el botón, esto por medio de una interrupción por flanco de subida asociada al GPIO, e incrementar una variable global de 32 bits cada vez que esto ocurra. Leerá la señal proveniente del botón utilizando el GPIO 15.

(a) Explique qué problema tendrá con las características de la señal proveniente del botón y qué componente debería poner entre el botón y microcontrolador para obtener el resultado esperado. (4)

> **Solución:** Al apretar el botón la señal no pasará de 0 a 1 inmediatamente, oscilará alrededor del voltaje asociado al 1 lógico antes de estabilizarse. Para solucionar esto podemos poner un Debouncer que estabilizará la señal.

(b) Escriba una subrutina que configure el GPIO 15 como una entrada con la interrupción por flanco de subida habilitada, e inicialice en 0 la variable global del contador. Respete el estándar de llamadas de RISC-V. (8)

Solución:

```
.data

 contador .space 4        # variable global de 32 bits

.text

 subrutina:
        li t0, 0x10060000 # dirección base
        li t1, 0x00008000 # 16° bit = 1
        sw t1, 0x04(t0)   # entrada
        sw zero, 0x08(t0) # salida
        sw t1, 0x018(t0)  # int. por flanco de subida
        sw zero, 0x20(t0) # int. por flanco de bajada
        la t2, contador   # |
        sw zero, 0(t2)    # | contador = 0
        ret
```

(c) Escriba la ISR que se encargará de incrementar la variable global definida anteriormente. Asuma que esta ISR ya está referenciada en el vector de interrupciones correspondiente y utilice la instrucción mret para finalizarla. (8)

Solución:

```
isr:
        addi sp, sp, -8   # Reservar 32bits x 2
        sw t0, 0(sp)      # Respaldar registro t0
        sw t1, 4(sp)      # Respaldar registro t1
        la t0, contador   # |
        lw t1, 0(t0)      # |
        addi t1, t1, 1    # |
        sw t1, 0(t0)      # | contador = contador + 1
        lw t1, 4(sp)      # Restaurar registro t1
        lw t0, 0(sp)      # Restaurar registro t0
        addi sp, sp, 8    # Restaurar stack
        mret
```