3.1.1. Una máquina que utiliza un reloj de 50 MHz, dispone de un sistema de memoria de dos niveles. La tasa de aciertos en el primer nivel es del 95% y la penalización total por fallo es de 20 ciclos de reloj. El tiempo de acceso al primer nivel es de un ciclo de reloj. Si se duplica el tamaño de la unidad del primer nivel, se mejora la tasa de fallos a 0,03 fallos por referencia a memoria. Este aumento de tamaño hace que el tiempo de acceso a la unidad mejorada aumente en un 20%. Indicar si la mejora es positiva, utilizando como parámetro en ambos casos el tiempo de acceso medio.

SOLUCIÓN:

- tiempo medio de acceso antes de la mejora 2 ciclos
- tiempo medio de acceso antes de la mejora 1,8 ciclos

La mejora es positiva y aumenta el rendimiento en un 11,11%.

3.1.2. En la tabla adjunta se indica el porcentaje de cada tipo de instrucciones y su duración media de un determinado sistema.

TIPO	ALU-INT	ALU-CF	Load/Store	OTRAS
PORCENTAJE	45%	25%	15%	15%
CICLOS DE EJECUCION	1	6	4	2

Además todas las instrucciones necesitan 4 ciclos para el acceso a memoria correspondiente a la captura de instrucción.
Se proponen dos mejoras independientes:
1.- Optimizar la ALU para coma flotante y reducir a la mitad el tiempo de ejecución.
2.- Incorporar una caché unificada con un tiempo de acceso de 1 ciclo, mejorando así tanto el tiempo de captura de instrucciones como la ejecución de Load/Store. La caché presenta una tasa de aciertos del 95% y un penalización total por cada fallo de 10 ciclos.
Se pide, utilizando necesariamente la ley de Amdahl:

a) ¿Cuál de las dos mejoras es la más eficiente y en cuánto?
b) ¿Cuál será la mejora total si se aplican ambas al tiempo?

SOLUCIÓN:

a) Mejora ALU-CF: $A_m = 2$; $F_m = (0,25 \times 6) / (0,45 \times 5 + 0,25 \times 10 + 0,15 \times 8 + 0,15 \times 6) = 1,5 / 6,85 = 0,219$

$A_G = 1 / [(1-0,219) + (0,219/2)] = 1,12$

Mejora caché: $T_{SIN} = 4 + 0,15 * 4 = 4,6$ ciclos. $T_{CON} = \%Acc \{t_c + (1-H)\ tp\} = 1,15 \{1 + 0,05x10\} = 1,725$ ciclos

$A_m = 2,667$;

$F_m = (0,45x4 + 0,25x4 + 0,15x8 + 0,15x4) / (0,45x5 + 0,25x10 + 0,15x8 + 0,15x 6) = 4,6 / 6,85 = 0,671$

$A_G = 1 / [(1-0,671) + (0,671/2,667)] = 1,72$

Es mejor la segunda mejora en un (1,72/1,12 =1,53) 53 %.

b) Mejora Global: Mejora ALU-CF: $A_G = 2,12$

3.1.3. Los parámetros que definen un sistema de memoria son: memoria principal de 32KB, memoria caché completamente asociativa de 4KB, tamaño de bloque de la caché, 8 palabras (16 bytes). El tiempo de acceso a la memoria principal es 10 veces mayor que el de la memoria caché. Se pide:

a) ¿Cuántos comparadores de hardware son necesarios?
b) ¿Cuál es el tamaño del campo para la etiqueta?
c) Si se utilizara correspondencia directa, ¿cuál sería el campo de la etiqueta?
e) Si el tiempo de acceso a caché es de 200 ns, ¿cuál será la razón de aciertos necesaria para lograr un tiempo de acceso medio de 500 ns?

SOLUCIÓN:

a) Son necesarios 256 comparadores de 11 bits.

b) El campo de la etiqueta es de 11 bits.

c) El campo de la etiqueta es ahora de 3 bits.

e) H ($t_B = t_m$)= 85%. H ($t_B = 8 \cdot t_m$) = 98,1%.

3.1.4. La siguiente secuencia de números en decimal, representa una cadena de direcciones de 8 bits para el acceso a memoria: 1, 4, 8, 5, 23, 20, 17, 19, 23, 56, 9, 11, 4, 43, 5, 6, 9 y 17. Suponiendo una caché de correspondencia directa con 8 bloques de una palabra, que inicialmente está vacía, indicar para cada referencia de la lista si corresponde a un acierto o a un fallo. Indicar el valor en binario de las etiquetas almacenadas en caché al final de la secuencia. Suponer 1 palabra = 1 byte.

SOLUCIÓN:

a)

Sec	1	4	8	5	23	20	17	19	23	56	9	11	4	43	5	6	9	17
A/F	F	F	F	F	F	F	F	F	A	F	F	F	F	F	A	F	A	F

b)

ENTRADA	**ETIQUETA**	**DATO**
000	00111	56
001	00010	17
010	- - - -	
011	00101	43
100	00000	4
101	00000	5
110	00000	6
111	00010	23

3.1.5. computador tiene una memoria principal de 16 Mbytes y emplea un bus de datos de 32 bits, dispone de una unidad caché de 8 kBytes. Determinar el número de bits de cada campo de la dirección según las siguientes organizaciones:

a) Correspondencia directa con una palabra por bloque.
b) Correspondencia directa con ocho palabras por bloque.
c) Correspondencia asociativa por conjuntos de cuatro vías y dos palabras por bloque.
d) Completamente asociativa y cuatro palabras por bloque.

SOLUCIÓN:

Memoria de 16 MB, es decir: 24 bits dirección

Bus de datos de 32 bits, es decir 4 bytes / palabra

Caché de 8 kB = 2^{13} Bytes

a) Correspondencia directa 4 bytes/bloque: $2^{13}/2^2 = 2^{11}$ entradas

ETIQUETA	**INDICE**	**BYTE/BLOQUE**
11 bits	11 bits	2 bits

b) Correspondencia directa con 32 bytes/bloque: $2^{13}/2^5 = 2^8$ entradas

ETIQUETA	**INDICE**	**BYTE/BLOQUE**
11 bits	8 bits	5 bits

c) Correspondencia asociativa por conjuntos, 4 vías y 8 bytes/bloque: $2^{13}/2^3 2^2 = 2^8$ entradas

ETIQUETA	**INDICE**	**BYTE/BLOQUE**
13 bits	8 bits	3 bits

d) Completamente asociativa y 16 bytes/bloque:

ETIQUETA	**BYTE/BLOQUE**
20 bits	4 bits

3.1.6. Una memoria caché de correspondencia directa, tiene un total de 256 bloques, de 16 bytes cada uno. El procesador genera direcciones de 32 bits. Se pide:
a) Indicar los bits de los campos en los que se divide la dirección.
b) Indicar el número de fallos que se generan, cuando se solicitan desde la CPU un bucle que se ejecuta diez veces con la siguiente secuencia de direcciones (en hexadecimal): A3272105 -- A3502120 -- A3271130 -- A3272103 -- A3502104 -- A3261130 y A3272104.
NOTA: La caché está llena, pero inicialmente no contiene ninguna de las direcciones de la secuencia anterior.

SOLUCIÓN:
a)

ETIQUETA	INDICE	BYTE/BLOQUE
20 bits	8 bits	4 bits

b) Primera vez que se ejecuta el bucle:

	INDICE	ETIQUETA	DIRECCIONES EN CACHE	A/F
A3272105	10	A3272	A3272100,.....,A327210F	F
A3502120	12	A3502	A3502120,.....,A350212F	F
A3271130	13	A3271	A3271130,.....,A327113F	F
A3272103	10	A3272	A3272100,.....,A327210F	A
A3502104	10	A3502	A3502100,.....,A350210F	F
A3261130	13	A3261	A3261130,.....,A326113F	F
A3272104	10	A3272	A3272100,.....,A327210F	F

Las nueve veces restantes que se ejecuta el bucle:

	INDICE	ETIQUETA	DIRECCIONES EN CACHE	A/F
A3272105	10	A3272	A3272100,.....,A327210F	A
A3502120	12	A3502	A3502120,.....,A350212F	A
A3271130	13	A3271	A3271130,.....,A327113F	F
A3272103	10	A3272	A3272100,.....,A327210F	A
A3502104	10	A3502	A3502100,.....,A350210F	F
A3261130	13	A3261	A3261130,.....,A326113F	F
A3272104	10	A3272	A3272100,.....,A327210F	F

Total de fallos: 6 + 9x4 = 42 fallos.

3.1.7. Una caché asociativa por conjuntos tiene un tamaño de bloque de cuatro palabras de 16 bits y un tamaño de conjunto de 2. La caché puede acomodar un total de 4096 palabras. El tamaño de memoria principal que es transferible a caché es de 64K x 32 bits.
Diseñar la estructura de la caché y mostrar como son interpretadas las direcciones del procesador.

SOLUCIÓN:

a) Los campos en los que se divide la dirección del procesador para la caché de correspondencia directa son:

ETIQUETA	INDICE	BYTE/BLOQUE
6 bits	9 bits	3 bits

La unidad caché dispone de 512 entradas que habilitan dos bloques uno por cada vía. Cada bloque tiene 6 bits de etiqueta para la comprobación, 4 palabras de 2 bytes, un bit de modificación y cuatro bits de validez uno por cada palabra.

3.1.8. Considérese que un programa accede a las siguientes direcciones de memoria (en decimal) 1, 4, 8, 5, 20, 17, 19, 56, 9, 11, 4, 43, 5, 6, 9,17
Cada una de estas referencias accede a datos de 1 byte de tamaño. Indicar en todos los casos el contenido final del directorio caché y el número total de fallos. En todos los casos se emplea una política de ubicación LRU. Suponga que se tiene una caché de 16 bytes de capacidad, inicialmente vacía.
a) Con bloques de 1 byte y de correspondencia directa.
b) Con bloques de 4 bytes y de correspondencia directa.
c) Con bloques de 1 byte y asociativa por conjuntos de dos vías.
d) Con bloques de 4 bytes y totalmente asociativa.
e) Con los datos obtenidos indicar que estructura caché es la mejor, cual la peor y cual es el porcentaje de mejora entre ambas organizaciones.

SOLUCIÓN:

Fallos/llamadas: **a)** 13/16, **b)** 10/16, **c)** 12/16 y **d)** 10/16.

e) Por tanto la mejor es la caché D y la peor la caché A con una diferencia de 6/3 = 2,0 (100 %)

3.1.9. Considerar una unidad caché que contiene 512 bytes de información. La unidad es asociativa por bloques de 4 vías con 4 palabras por bloque. La unidad dispone de los bits de validez y de los bits de modificación (suciedad) necesarios. El bus de direcciones es de 32 bits mientras que el de datos es sólo de 16 bits. Se pide:

a) ¿Cómo se divide la dirección para el acceso a la caché?

b) ¿Cuál es el tamaño en bits de la unidad caché?

c) ¿Cuál es el orden (índice) de la línea de entrada en la que se debe guardar la información correspondiente a la dirección $000010AF_{16}$?

d) La dirección de memoria anterior y la dirección $FFFF7AXY_{16}$ se encuentran simultáneamente guardadas en la unidad caché. Ambas tienen el mismo índice, señalan a la misma palabra dentro del bloque pero se encuentran en distinta vía. ¿Cuántas direcciones cumplen las condiciones anteriores? ¿Cuáles son los posibles valores de X e Y.

SOLUCIÓN:

a) Etiqueta: 25 bits. Indice: 4 bits. Bytes en bloque: 3 bits.

b) Considerando 1 bit de validez por palabra (4 por bloque) y 1 bit de modificación por bloque =>

Tamaño: 6.016 bits.

Considerando 1 bit de validez y 1 bit de modificación por bloque =>

Tamaño: 5.824 bits..

c) El índice de la dirección $000010AF_{16}$ señala a la posición 6ª de la tabla.

d) Son 4 las direcciones posibles: FFFF7A2E, FFFF7A2F, FFFF7AAE y FFFF7AAF.

3.1.10. La siguiente figura representa los contenidos de una caché con 4 Vías, un tamaño de 64 bytes y política de funcionamiento de Escritura Directa sin asignación en escritura (EDSAE) y reemplazamiento LRU.

VIA0			VIA1			VIA2			VIA3		
Etiqueta	**Word1**	**Word0**	**Etiqueta**	**Word1**	**Word0**	**Etiqueta**	**Word1**	**Word0**	**Etiqueta**	**Word1**	**Word0**
1101	A1	A0	0001	A3	A2	1110	A5	A4	1111	AF	AE
0011	B1	B0	0001	B3	B2	0111	B5	B4			
1110	C1	C0	1010	C3	C2						
1110	D1	D0	1101	D3	D2	1111	D5	D4	0000	D7	D6
0001	E1	E0									
0100	F1	F0	1111	F3	F2						

La etiqueta está expresada en binario y los campos Word1 y Word0 están expresados en hexadecimal y se corresponden respectivamente con la palabra alta y baja de cada bloque.

a) Indique todos los recursos necesarios para su implementación.

b) ¿Qué dirección de memoria corresponde a las siguientes palabras que se encuentran almacenadas en la caché?

Dato	Dirección
C3	
A0	
A3	

c) Suponga que se realiza la secuencia de accesos de la tabla, en la cual la primera columna indica si el acceso es una lectura o una escritura. Suponga que al iniciar la secuencia, la vía LRU en todas las entradas corresponde a los datos guardaos en la vía 3. Rellene las columnas con los valores correspondientes, indicando si hay cambio en la caché, el lugar donde se produce.

R/W	Dirección	Vía	Número de entrada	Acierto/Fallo
Lectura	11100110			
Escritura	11110011			
Lectura	11110010			
Escritura	11110011			
Lectura	00000010			

d) Partiendo de situación mostrada en la figura, indique la menor secuencia de accesos a memoria que complete la última entrada de todas las vías.

e) Se modifica la estructura de la caché incrementando en 3 bits el campo de etiqueta y se mantiene en 64 bytes su capacidad total con el mismo número de bytes por bloque. ¿Cuál es la organización de la nueva caché?

SOLUCIÓN:

a) Recursos necesarios para su implementación.

Directorio: 4 vias x 8 entradas(4bit tag +1bitV+ 2bitLRU) = 224 bits = 28 bytes y 4 comparadores de 4 bits

Datos: 4 vias x 8 entradas(16bits)= 512 bits = 64 bytes.

b)

Dato	**Dirección**
C3	**10100111**
A0	**11010000**
A3	**00010001**

c)

R/W	Dirección	Vía	Número de entrada	Acierto/Fallo
Lectura	11100110	**0**	**3**	**A**
Escritura	11110011	**-**	**-**	**F**
Lectura	11110010	**3**	**1**	**F**
Escritura	11110011	**3**	**1**	**A**
Lectura	00000010	**0,1,2**	**1**	**F**

d) Por ejemplo 2 lecturas de las direcciones: 00001110, 10001110

e) La nueva caché es completamente asociativa con 32 entradas

3.1.11. Se tiene un sencillo sistema RISC donde el bus de datos y de direcciones son ambos de 16 bits. El sistema dispone de una unidad caché no unificada. Ambas cachés tienen una capacidad de 16 KB, tienen una estructura asociativa de 4 vías, con una política LRU para el reemplazamiento, guardan 8 palabras por bloque y existen bits de suciedad o modificación donde corresponda. Se sabe que el acceso a memoria DRAM es de 100 ns y el acceso a caché es de 10 ns. Inicialmente ambas cachés están vacías. Justificando brevemente las respuestas, se pide:

a) Los campos en los que se divide la dirección para el acceso a las cachés.
b) La tasa de aciertos en la caché de instrucciones.
c) La tasa de aciertos en la caché de datos
d) El tiempo total utilizado para el acceso a memoria en la ejecución del código mostrado.

Nota: El código suma 100 a un vector de 50 elementos de 2 bytes, ubicado a partir de la posición 4400_{16} y lo reescribe modificado en la misma posición.

Dirección		**Instrucción**	
00EC		MOVE 0,R1	; 0 ⇒ R1
00EE		MOVE 50,R2	; 50 ⇒ R2
00F0	L1:	SUB R2,1,R2	; R2 -1 ⇒ R2
00F2		Bneg Fin	; Salto condicional. Si N = 1, salta a Fin
00F4		LOAD #4400, R1, R4	; Mem[4400+(R1)] ⇒ R4
00F6		ADD R1, 2, R1	; R1+ 2 ⇒ R1
00F8		CALL Sub	; Llamada a la rutina Sub
00FA		STORE R6, #4400, R1	; R6 ⇒ Mem[4400+(R1)]
00FC		JUMP L1	; Salta incondicional a la etiqueta L1
00FE	Fin:	STOP	; Fin del programa principal
…..		……	
0400	Sub:	ADD R4,100,R6	; R4 + 100 ⇒ R6
0402		RETURN	; Vuelve de la rutina

SOLUCIÓN:

a) Analizando los datos del enunciado se obtiene que:

2^{14} bytes / 2^2 vias = 2^{12} bytes/vía.

2^{12} bytes/via / 2^4 bytes/bloque = 2^8 bloques/via = 2^8 entradas.

Los campos en los que se divide la dirección son:

Etiqueta	Índice	Bytes/bloque
4	8	4

b) El número de accesos a la memoria de instrucciones es:

2 (instr. iniciales)+ 50 x 7 (bucle principal) + 50 x 2 (bucle rutina) + 3 (instr. finales) = 455 accesos

Hay 3 fallos que ocurren en las direcciones 00EC, 00F0 y 0400, que llenan la primera vez los índices en hexadecimal 0E, 0F y 40.

En cada fallo, se carga un bloque completo que luego suponen los 452 aciertos.

Hay por tanto 3 fallos en 455 llamadas es decir la tasa de aciertos es: 99,34%

c) El número de accesos a la memoria de datos es: 50 (lecturas) + 50 (escrituras) = 100 accesos.

Hay 7 fallos que ocurren en las direcciones 4400, 4410, 4420, 4430, 4440, 4450 y 4460, que llenan la primera vez los índices en hexadecimal 40, 41, 42, 43, 44, 45 y 46.

En cada fallo, se carga un bloque completo que luego suponen 43 aciertos en las lecturas y 50 aciertos en las escrituras.

La presencia de los bits de modificación indican que se trata de una caché de post-escritura. Durante la ejecución del programa, no se precisa ningún reemplazamiento, por tanto se considera que $w_M = 0$.

Hay por tanto 7 fallos en 100 llamadas es decir la tasa de aciertos es: 93%

d) La ecuación que calcula el tiempo total viene dada por:

$T^{INSTRUCCIONES} = Nº\_Acc^{INS} \times \{ t_C + (1\text{-}H)\, t_B \} = 455 \{ 0{,}01 + (0{,}0066) \times (8x0{,}1) \} = 6{,}95$ msec

$T^{DATOS} = Nº\_Acc^{DAT} \times \{ t_C + (1\text{-}H)\, t_B \} = 100 \{ 0{,}1 + (0{,}07) \times (8x0{,}1) \} = 6{,}60$ msec

$T^{TOTAL} = T^{INSTRUCCIONES} + T^{DATOS} = 6{,}95 + 6{,}60 = 13{,}55$ msec

3.1.12. Un cierto sistema dispone de un bus de direcciones de 32 bits, siendo el tamaño de la palabra de un byte. Se la desea dotar de una pequeña unidad caché para instrucciones, con las siguientes características: 16 bloques, siendo cada bloque de 4 bytes. No son necesarios los bits de validez y la estrategia para reemplazamiento es LRU. Se consideran dos diferentes opciones, una caché de correspondencia directa (caché A) y una completamente asociativa (caché B).
Mostrar en un dibujo los bloques en los que se divide la dirección de memoria para el acceso a memoria en cada una de las opciones de la caché.
Dar una expresión matemática que a partir de la dirección de memoria n (n>=0), calcule la entrada de acceso a la unidad caché A.
Asumir que inicialmente la caché esta vacía, y que el tamaño de instrucción es de 1 byte. Suponer que se lleva a cabo la siguiente secuencia de instrucciones: Las direcciones de memoria de 0-63 se ejecutan una vez. Las direcciones de memoria de la 64-131, pertenecen a un bucle que se ejecuta 10 veces.
Encontrar el número de faltas en ambas cachés. Calcular el tiempo de acceso medio en ambos casos y señalar para este programa cuál es la caché más rentable y en qué porcentaje se valora la mejora. Se conoce que tiempo de acceso para ambas cachés es t_{CACHE} =20 ns y el tiempo de penalización por fallo en ambos casos es t_B=200 ns.

SOLUCIÓN:

a) Los campos en los que se divide la dirección del procesador para la caché de correspondencia directa son:

ETIQUETA	INDICE	BYTE/BLOQUE
26 bits	4 bits	2 bits

Los campos en los que se divide la dirección del procesador para la caché completamente asociativa son:

ETIQUETA	BYTE/BLOQUE
30 bits	2 bits

b) Una expresión matemática como n = (n/4) MOD 16 indica el valor del índice de acceso a la caché.

c) 1) Caché de correspondencia directa: se producen 51 faltas sobre 744 llamadas. Tasa de fallos del 6,85%

2) Caché completamente asociativa: se producen 186 faltas sobre 744 llamadas. Tasa de fallos del 25%

d) La caché de correspondencia directa es más rentable en un 107%.

3.1.13. Considerar tres máquinas con diferentes configuraciones de memoria caché:

MAQ1: Correspondencia directa con bloques de 1 palabra. La tasa de fallos es del 4% para instrucciones y del 8% para datos.
MAQ2: Correspondencia directa con bloques de 4 palabras. La tasa de fallos es del 2% para instrucciones y del 5% para datos.
MAQ3: Asociativa por conjuntos de dos vías con bloques de 4 palabras. La tasa de fallos es del 2% para instrucciones y del 4% para datos.
Para estas máquinas, la mitad de las instrucciones contienen una referencia de datos. Suponer que la penalización, en ciclos, de fallos de la caché es 6 más el tamaño del bloque en palabras. El CPI suponiendo máquinas ideales (sin fallos) vale 2 en todos los casos.
a) Determinar que máquina emplea más ciclos en los fallos de la caché.
b) Si las frecuencias de funcionamiento son 100 MHz para MAQ1 y MAQ2 y 80 MHz para MAQ3, determinar el rendimiento de la máquina más rápida respecto a la más lenta.

SOLUCIÓN:

a) MAQ1 emplea 0,56 ciclos, MAQ2 emplea 0,45 ciclos y MAQ3 emplea 0,40 ciclos.

b) Si se tiene en consideración la frecuencia de operación, la MAQ2 es un 22,5% más eficiente que la MAQ3 y un 4,5% más eficiente que la MAQ1.

3.1.14. Se tiene un sencillo sistema RISC donde el bus de datos es de 16 bits y el de direcciones de 12 bits. Todas las instrucciones ocupan una palabra. El sistema dispone de una unidad caché no unificada. Las cachés tienen una capacidad de 32 Bytes.
La caché de instrucciones tiene una estructura asociativa de 2 vías, con una política LRU para el reemplazamiento. La caché de datos es de correspondencia directa (CD). Ambas cachés guardan dos palabras por bloque. No existen bits de control de ningún tipo. Para cada palabra, el byte menos significativo se almacena en la posición de memoria más baja. Dentro de un bloque la palabra de la izquierda corresponde a una posición de memoria superior respecto a la de la derecha
Justificando necesaria y brevemente las respuestas, se pide:
a) Los campos en los que se divide la dirección para el acceso a las cachés.
b) El estado final de las memorias cachés tras la ejecución del programa. Suponer las cachés vacías y que la LRU en caso de ser necesaria es la vía 1. Para indicar el contenido de la memoria caché utilizar la notación D(XYZ) para indicar la palabra contenida en la dirección XYZ. Por ejemplo D(1EA) es el contenido de la primera instrucción del programa. Las lecturas de memoria principal son por bloques completos.
c) La tasa de aciertos en la caché de instrucciones y de datos.
Nota: El código realiza la suma de los 10 elementos de un vector (con datos de 2 bytes), ubicado a partir de la posición 402_{16} y queda el resultado en el Registro R3.

1EA	MOVE 0,R1	;0 ⇒ R1, apuntador al arreglo
1EC	MOVE 10,R2	;10 ⇒ R2, cuenta las iteraciones
1EE	MOVE 0,R3	;0 ⇒ R3, lleva la suma
1F0 L1:	SUB R2,1,R2	;R2-1 ⇒ R2
1F2	Bneg Fin	;Salto condicional. Si N = -1, salta a Fin
1F4	LOAD #402,R1,R4	;Mem[402+(R1)] ⇒ R4
1F6	ADD R1,2,R1	;R1+2 ⇒ R1
1F8	ADD R3,R3,R4	;R3+R4 ⇒ R3
1FA	JUMP L1	;Salta incondicional a la etiqueta L1
1FC	Fin: STOP	;Fin del programa

SOLUCIÓN:
a)
Los campos en los que se divide la dirección para el acceso a instrucciones son:

TAG	IND	B/B
8	2	2

4 bytes por bloque ⇒ 2 bits en Byte/Bloque.
2^5 Bytes / 2^2 By/Bq * 2 vías ⇒ 2^2 Bq/vía ⇒ 2 bits índice.

Los campos en los que se divide la dirección para el acceso a datos son:

TAG	IND	B/B
7	3	2

2 palabras (4 bytes) por bloque ⇒ 2 bits en Byte/Bloque.
2^5 Bytes / 2^2 By/Bq ⇒ 2^3 Bq ⇒ 3 bits índice.

b) Caché de instrucciones (CA2V)

VIA 1			VIA 2		
Tag1	Datos		Tag1	Datos	
1F	D(1F2)	D(1F0)			
1F	D(1F6)	D(1F4)			
1E	D(1EA)	D(1E8)	1F	D(1FA)	D(1F8)
1E	D(1EE)	D(1EC)	1F	D(1FE)	D(1FC)

Caché de datos: CD, 2 pal./bloq.

Ind	Tag	Datos	
000	0100 000	D(402)	D(400)
001	0100 000	D(406)	D(404)
010	0100 000	D(40A)	D(408)
011	0100 000	D(40E)	D(40C)
100	0100 000	D(412)	D(410)
101	0100 000	D(416)	D(414)
110			
111			

c) Las tasas de acierto son respectivamente: 60/66 ⇒ 90,9% para instrucciones y4/10 ⇒ 40% para datos.

Las posiciones de memoria que contienen las instrucciones y los fallos (F) o aciertos (A) que se producen son: 1EA (1F), 1EC (1F), 1EE (1A), 1F0 (1F,10A), 1F2 (11A), 1F4 (1F,9A), 1F6 (10A), 1F8 (1F,9A), 1FA (10A) y 1FC (1F)

Las posiciones de memoria que contienen los datos y los fallos (F) o aciertos (A) que se producen son:

402 (F), 404 (F), 406 (A), 408 (F), 40A (A), 40C (F), 40E (A), 410 (F), 412 (A) y 414 (F)

3.1.15. La caché de un cierto sistema es unificada, asociativa de dos vías (CA2V), 16 entradas por vía y cuatro palabras por bloque. En un momento determinado, justo tras finalizar la instrucción Clear R2, la caché presenta el estado que se muestra en el esquema adjunto, donde se marcan con XXXX el dato que cada bloque contiene. Además asociado a cada bloque, se indican el estado del contador (1bit) utilizado para el algoritmo LRU implementado siguiendo el sistema estudiado en clase. A continuación se ejecuta el bucle 5 veces y se termina con la ejecución de la instrucción Halt. Nota: El sistema sólo maneja instrucciones y datos de 32 bits.

```
PC
0F00A4          Clear   R2              ; 0 → R2
0F00A8  Bucle:  Load    R1, R2, #7000   ; Mem [(R2) + 007000] → R1
0F00AC          Add     R3,R1,R5        ; R1 + R5 → R3
0F00B0          Add     R2, R2, #4      ; R2 + 4 → R2
0F00B4          Bne     R2, #20, Bucle  ; Si R2 ≠ 20₁₀, salta a bucle
0F00B8          Jmp     Fin             ; Salta a Fin
- - - - -
- - - - -
F00500  Fin:    Halt                    ;  Finaliza la ejecución
```

VIA 1			VIA 2		
DIRECTORIO	**BLOQUES**	**LRU**	**DIRECTORIO**	**BLOQUES**	**LRU**
0F00	XXXX	1	F000	XXXX	0
00F0	XXXX	0	0000	XXXX	1
000F	XXXX	0	000F	XXXX	1
0000	XXXX	1	00F0	XXXX	0
F000	XXXX	1	0F00	XXXX	0
0F00	XXXX	0	F000	XXXX	1
00F0	XXXX	0	0000	XXXX	1
000F	XXXX	1	000F	XXXX	0
0000	XXXX	1	00F0	XXXX	0
F000	XXXX	0	0F00	XXXX	1
0F00	XXXX	0	F000	XXXX	1
00F0	XXXX	1	0000	XXXX	0
000F	XXXX	1	000F	XXXX	0
0000	XXXX	0	00F0	XXXX	1
F000	XXXX	0	0F00	XXXX	1
0F00	XXXX	1	F000	XXXX	0

Se pide, con los datos de los que se dispone y justificando cada una de las respuestas:

a) Los campos en los que se divide la dirección para el acceso a la caché**.**

b) El tamaño de la caché, señalando los bytes de datos, los bits asociativos y los comparadores de los que dispone.

c) La tasa de aciertos para el código ejecutado

d) El estado de la caché tras la ejecución, incluyendo los contadores del algoritmo. Utilizando el esquema facilitado, señalar sólo aquellas entradas que hayan variado durante la ejecución, indicando el valor del índice de la entrada correspondiente y los valores para la etiqueta y contador LRU finales.

SOLUCIÓN:

a) 2^2 palabras/bloque * 2^2 bytes/palabra = 2^4 bytes por bloque. Se necesitan 4 bits para su selección.

Si la caché tiene 2^4 entradas por vía Se necesitan 4 bits para seleccionar una de las entradas.

Si como vemos en el código, las direcciones son de 24 bits, nos quedan 16 bits para la etiqueta.

ETIQUETA	INDICE	BYTE EN BLOQUE
16	4	4

b) a1) La caché guarda un total de 2 vías * 2^4 bloques/vía * 2^4 bytes/bloque = 2^9 bytes de datos.

a2) La caché dispone de un total de 2 vías * 2^4 entradas/vía * 2^4 bits/entrada = 2^9 bits asociativos y

2 vías * 2^4 entradas/vía * 1bit = = 2^5 bits para contadores LRU

a3) La caché necesita un total de 2 comparadores cada uno de 16 bits.

Se necesita un comparador por cada vía

c) A 4 direcciones de código, 0F00A8:0F00B4 accede 5 veces sólo falla la primera vez 0F00B0.

A 2 direcciones de código 0F00B8 y F00500, accede sólo 1 vez, sólo falla en la F00500.

A las 5 direcciones de datos 007000:007010, accede sólo 1 vez y falla en la 007000 y 007010.

Hay por tanto 27 accesos y 4 fallos. Es decir una tasa de aciertos del (23/27 = 0,85) 85%.

Dirección de acceso:

ETQ	IND	B/B	A/F	Explicación:
0F00	A	8	A	Acierto, índice A, vía 1.
0070	0	0	F	Fallo. Carga un bloque índice 0, vía 1 (LRU). Cambian los contadores de ambas vías.
0F00	A	C	A	Acierto, índice A, vía 1.
0F00	B	0	F	Fallo. Carga un bloque índice B, vía 1 (LRU). Cambian los contadores de ambas vías.
0F00	B	4	A	Acierto, índice B, vía 1.
0F00A8:0F00B4			A	16 aciertos en el acceso a código de las 4 iteraciones.
007004:00700C			A	3 aciertos en el acceso a datos, índice 0, vía 1. 3 iteraciones
0070	1	0	F	Fallo. Carga un bloque índice 1, vía 2 (LRU). Última iteración
0F00	B	8	A	Acierto, índice B, vía 1.
F005	0	0	F	Fallo. Carga un bloque índice 0, vía 2 (LRU). Cambian los contadores de ambas vías.

d)

INDICE	VIA 1			VIA 2		
	DIRECTORIO	BLOQUES	LRU	DIRECTORIO	BLOQUES	LRU
0	**0070**	XXXX	**1**	**F005**	XXXX	**0**
1	00F0	XXXX	**1**	**0070**	XXXX	**0**
B	**0F00**	XXXX	**0**	0000	XXXX	**1**

3.1.16. El siguiente programa escrito en C se ejecuta en un procesador que dispone de una caché de datos con 8 palabras por bloque (32 bytes) y un tamaño de 256 bytes.

```
int i, j, c, inc, array[512];
...
 for ( i=0 ; i<1000; i++)
    for (j=0; j<512 ; j=j+inc)
        c = array[ j ] + 17;
…
```

Sabiendo que un entero ocupa una palabra, que la caché está inicialmente vacía y considerando que toda la actividad de la caché se genera en los accesos a memoria de la variable array, se pide **justificando necesariamente la respuesta** cual es la tasa de fallos esperada en el acceso a la caché si:

a) La caché es de correspondencia directa (CD) y la variable inc = 256.
b) La caché es de correspondencia directa y la variable inc = 255.
c) Que ocurre en ambos casos anteriores si la caché es asociativa de 2 vías (CA2V).

Nota: Cualquier respuesta sin su correspondiente explicación no será calificada aunque sea correcta.

SOLUCIÓN:

Sea cual sea el tamaño de la dirección, se utilizan los 3 LSB bits de la misma para señalar el byte en el bloque.

Si el tamaño de la cache es de 256 (2^8) bytes y el tamaño de un bloque es 32 (2^5) bytes, tenemos un total de 8 (2^3) bloques. Por tanto se necesitan 3 bits para indexar la caché de correspondencia directa y 2 bits para indexar la caché asociativa de 2 vías.

	Etiqueta	Índice	Byte/bloque
CD	-----	3 bits	5 bits
CA2V	-----	2 bits	5 bits

Para la caché CD, cada 256 (2^8) bytes se repite el mismo índice. Es decir, dada una dirección el valor del índice que ocupa en la caché se calcula operando INDICE = (DIR/2^5) mod 8.

Para la caché CA2V, cada 128 (2^7) bytes se repite el mismo índice. Es decir, dada una dirección el valor del índice que ocupa en la caché se calcula operando INDICE = (DIR/2^5) mod 4

a) Si la variable inc = 256, para cada i, hay sólo 2 iteraciones en j, j = 0, j = 256. Entre estas dos llamadas a la memoria de datos, hay 256*4 = 1024 bytes de diferencia. Ambas direcciones ocuparán con diferentes etiquetas, el mismo índice y por tanto en la primera iteración (i=0) habrá dos fallos y el segunda acceso ocupará la misma posición que el primero. Para cada una de las 999 iteraciones restantes, los accesos a las mismas posiciones de memoria que en la primera iteración, producirá dos fallos cada vez. Por tanto, cada acceso a memoria supone un fallo en la caché, luego…

…la tasa de fallos es del (2000/2000) 100%.

b) Si la variable inc = 255, para cada i, hay 3 iteraciones en j, j = 0, j = 255 y j = 510. La distancia en bytes es 255*4 = 1020 bytes entre cada una de las direcciones. Dos de estas direcciones ocuparán el mismo índice en la caché** y por tanto en la primera iteración (i=0) habrá 3 fallos y para cada una de las 999 restantes, 2 fallos y 1 acierto, luego…

…la tasa de fallos es del (2001/3000) 66,7%.

c) Para la caché CA2V, ocurre lo mismo que en el caso CD aunque haya menos entradas. A diferencia del caso anterior, esta caché permite guardar al tiempo 2 bloques con el mismo índice. Por tantoi para la primera iteración, al estar la caché vacía se producirán 2 y 3 fallos para cada valor de inc, pero en las 999 iteraciones de i restantes, siempre se produce un acierto. Por tanto…

…la tasa de fallos es del (2/2000) ~0,01% si inc = 256.

…la tasa de fallos es del (3/3000) ~0,01% si inc = 255.

**** [N/8] mod 8 = k;**

[{N+2040}/8] mod 8 = [{N+(255*8)}] mod 8 = [{N/8+255}] mod 8 = [N/8] mod 8 + [255] mod 8 = k + 7

[{N+1020}/8] mod 8 = [{N+(127*8)+4}/8] mod 8 = [(N+4)/8+127] mod 8 = [(N+4)/8] mod 8 + [127] mod 8 =

$$= [(N+4)/8] \bmod 8 + 7 = \begin{cases} k + 7 \\ (k+1) + 7 = k + 8 = k \end{cases}$$

3.1.17. El microprocesador Microblaze es un procesador RISC de 32 bits (palabras, direcciones de memoria y ruta de datos de 32 bits) segmentado con 5 etapas destinado a sistemas empotrados basados en lógica programable. Posee una estructura MMU y cache de datos configurable en tamaño. Las características de las caché son:
- Correspondencia directa (Asociativo de una vía).
- Escritura directa (Write Through), con buffer de escritura (en caso de ser necesario).
- Tamaño de la caché y de la etiqueta configurable.
- El tamaño del bloque en caché se puede configurar como 4 ú 8 palabras.

El sistema permite "cachear" un cierto rango de posiciones consecutivas a través de la configuración de dos registros (CACHE_BASEADDR y CACHE_HIGHADDR). El resto de las direcciones quedan sin cachear. Existe un único bit de control que es el de validez que se utiliza tanto en la inicialización como para invalidar un bloque de datos por el software. En caso de generarse una dirección en el espacio "no cacheable" el sistema caché descarta la petición y se trae el dato de la memoria principal sin que la caché actúe.

Suponga que se configura un sistema con una caché de instrucciones que tiene un total de 16KB organizado en bloques de 4 palabras y una caché de datos de 64KB en bloques de 8 palabras y que los registros de direcciones "cacheables" son:

C_ICACHE_BASEADDR= 0x00300000 y C_ICACHE_HIGHADDR=0x0030FFFF (Caché Instrucciones).

C_DCACHE_BASEADDR= 0x00400000 y C_DCACHE_HIGHADDR=0x0043FFFF (Caché Datos).
Se pide:

a) Tamaño de memoria de datos e instrucciones que se han de cachear

Tamaño mem Instrucciones cacheables:
0x0030FFFF - 0x00300000 + 1 = 10000 hex = 2**16 bytes = 65536 bytes = 64 KB.

Tamaño mem Instrucciones cacheables:
0x0043FFFF - 0x00400000 + 1 = 40000 hex = 2**18 bytes = 262144 bytes = 256 KB

b) Considerando sólo el rango de direcciones cacheable, señalar la forma en que se divide la dirección para el acceso a caché de instrucciones.

Bloques de 4 palabras de 4 bytes = 16 Bytes/bloque => 4 bits BB (Byte in Block)
Si se tiene 16KB de caché (2**14) y el tamaño del bloques de 2*4 bytes se usan 10 bits de índice.
Como la memoria que se cachea es de 16 bits y se usan 10 de índice y 4 de byte in bloque.
16-10-4 = 2 bits de etiqueta:

Los 32 bits: No se usan (16), Etiqueta (2), Índice (10), BB(4)

c) Considerando sólo el rango de direcciones cacheable, señalar la forma en que se divide la dirección para el acceso a caché de Datos.

Bloques de 8 palabras de 4 bytes = 32 Bytes/bloque => 5 bits BB (Byte in Block)
Si se tiene 64KB de caché (2**16) y el tamaño del bloques de 2*5 bytes, por tanto se usan 11 bits de índice.
Como la memoria que se cachea es de 18 bits y se usan 11 de índice y 5 de byte in bloque.
18-11-5 = 2 bits de etiqueta:

Los 32 bits: No se usan (14), Etiqueta (2), Índice (11), BB(5)

d) Tamaño total de las memorias caché, cantidad y tamaño de comparadores necesarios.

Mem Instrucciones: 2**10 entradas * (16 Bytes/bloque + 2 bits etiqueta + 1 bit validez) =
131 * 2**10 bits = 134144 bits = 16768 bytes = 16,375 KB.
Un comparador de dos bits para la etiqueta.

Mem Datos: 2**11 entradas * (32 Bytes/bloque + 2 bits etiqueta + 1 bit validez) =
259 * 2**11 bits = 530432 bits = 66304 bytes = 64,75 KB
Un comparador de dos bits para la etiqueta.

e) Si no se utiliza la caché de datos el sistema necesita de 3 ciclos de reloj para acceder a la memoria de datos tanto en lectura como escritura. La memoria de instrucciones es independiente (se puede acceder en paralelo) y posee un buffer de pre-captura que hace que el tiempo medio de acceso sea de 1,2 ciclos. Mientras el sistema accede a memoria el pipeline se detiene (*stall*). Suponga además que la cantidad de instrucciones **load** representan el 6 % y las de **store** el 4%. En la figura se observan los ciclos de *stall* que agrega el acceso a la memoria de datos, pero no se muestran las detenciones de la memoria de instrucciones.

	cycle 1	cycle 2	cycle 3	cycle 4	cycle 5	cycle 6	cycle 7	cycle 8	cycle 9
instruction 1	IF	OF	EX	MEM	WB				
instruction 2		IF	OF	EX	MEM	MEM	MEM	WB	
instruction 3			IF	OF	EX	Stall	Stall	MEM	WB

¿Qué CPI posee el sistema sin contar los riesgos de saltos?

CPI = CPI ideal + 0,2 ciclo Mem Instr + (6% load + 4% store)* 2 ciclos extras acceso mem datos =
= 1 + 0,2 + 0,1*2 = 1,4 ciclos

f) Suponga que se agrega una caché de instrucciones que tiene un tiempo de acceso en acierto de 1 ciclo y cuando falla, con una probabilidad del 3%, al margen de traer el bloque necesita un ciclo extra para actualizar las etiquetas. El tiempo de lectura-escritura es de 2 ciclos por cada palabra. También se agrega una caché de datos, la que tiene una probabilidad de acierto del 96% que tarda 1 ciclo en caso de acierto y en caso de fallo, tras traer el bloque, usa otro ciclo extra para actualizar las etiquetas. La memoria de datos necesita 3 ciclos para leer ó escribir cada palabra.

¿Qué CPI tiene el sistema ahora?

CPI = CPI ideal + CPI menInst + CPI menDatos

CPI memInst = Fallos* Penalización = 3% * (2 ciclos * 4 word/bloque + 1ciclo)
= 0,03*9 = 0,27 ciclos
CPI MemDatos = % accesos * Fallos* Penaliz = (6% load + 4% store)* 4% * (3 ciclos * 8 word/bloque + 1)
= 0,1 * 0,04 * 25 = 0,1 ciclos

CPI = 1 + 0,27 + 0,1 = 1,37 ciclos

3.1.18. Se pretende analizar y comparar el rendimiento de dos procesadores. El PROC_1, es un procesador de emisión simple segmentado en 5 etapas (como el visto en la teoría y práctica) que tiene las siguientes etapas: IF, ID, EX, MEM, WR. Este procesador funciona a una velocidad de reloj de 200 MHZ. El camino crítico en este sistema es el acceso a las memorias cachés (IF y MEM).

Para acelerar el cómputo se rediseña la segmentación para dividir los accesos a memoria en dos ciclos. De este modo el nuevo procesador (PROC_2) tiene 7 etapas (IF1, IF2, ID, EX, M1, M2, WR) y funciona a 250 MHz. Con esto el acceso a cachés implica enviar la dirección en el ciclo IF1 y recibir el resultado en el ciclo IF2 y esto se puede producir en cada ciclo de reloj (es decir en caso de acierto en caché no agrega demora). Lo mismo sucede para el acceso a la caché de datos en los ciclos M1, M2.

Considere que los procesadores ejecutan un código con la siguiente distribución en número de instrucciones:

Load	Store	Salto Cond	Salto Incod	Enteros
12%	8%	10%	2%	68%

En ambos procesadores la caché de instrucciones tiene una tasa de fallos de 2% y la de datos del 3%. El movimiento de los bloques de memoria a la caché implica 10 ciclos de reloj para la actualización de etiquetas y otras tareas en la caché de datos y de sólo 6 ciclos en la de instrucciones. Además cada transferencia implica 200 ns de tiempo de transferencia del bloque completo.

Se supone que todos los riesgos de datos se pueden adelantar y/o salvar por medio del compilador y que los riesgos de control en el PROC_1 penalizan 1 ciclo para los saltos incondicionales, 2 ciclos para los saltos efectivos y 0 ciclos para los saltos no efectivos.

Para el procesador segmentado en 7 etapas, en la etapa EX se conoce el resultado de la condición y el cálculo de la dirección se lleva a cabo en ID. El porcentaje de saltos condicionales (vale para ambos procesadores) con salto efectivo es del 33% y la estrategia de predicción de saltos de No efectiva.

a. ¿Qué sistema es más rápido y por cuánto? No tenga en cuenta el tiempo de llenado y vaciado del pipeline. Para la justificación de los ciclos de demora en los riesgos de control del PROC_2 utilice necesariamente las tablas adjuntas y responda en el sitio correspondiente.

b. Al segundo sistema se le rediseña el acceso a memoria principal agregando un segundo nivel de caché (L2). Con esto, las tasas de fallos locales en la caché de instrucciones son fi1 = 2% y fi2=5%, y para caché de datos fd1=3% y fd2=10% siendo la penalización por fallo en caché de 10 y 60 ciclos respectivamente (TB1 = 10 ciclos y TB2=60). ¿En qué porcentaje mejora al sistema PROC_2? Responda en el sitio correspondiente.

Tabla que justifica la demora adicional en **Salto incondicional**. Pierde: 2 ciclos

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
Jump	IF1	IF2	ID[1]	EX[2]	M1	M2	WR					
N		IF1	IF2	ID	EX	M1	M2	WR				
N+1			IF1	IF2	ID	EX	M1	M2	WR			
N+2												
…												
T				IF1	IF2	ID	EX	M1	M2			
T+1					IF1	IF2	ID	EX	M1			
T+2												

Tabla que justifica la demora adicional en **Salto condicional efectivo.** Pierde: 3 ciclos

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
Jump	IF1	IF2	ID[1]	EX[2]	M1	M2	WR					
N		IF1	IF2	ID	EX	M1	M2	WR				
N+1			IF1	IF2	ID	EX	M1	M2	WR			
N+2				IF1	IF2	ID	EX	M1	M2	WR		
…												
T					IF1	IF2	ID	EX	M1	M2		
T+1						IF1	IF2	ID	EX	M1		
T+2												

Tabla que justifica la demora adicional en **Salto condicional NO Efectivo**. Pierde: 0 ciclos

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
Jump	IF1	IF2	ID[1]	EX[2]	M1	M2	WR					
N		IF1	IF2	ID	EX	M1	M2	WR				
N+1			IF1	IF2	ID	EX	M1	M2	WR			
N+2				IF1	IF2	ID	EX	M1	M2	WR		
…												
T												
T+1												
T+2												

En (1) calcula dirección, en (2) resuelve condición

a. ¿Qué sistema es más rápido y por cuánto? No tenga en cuenta el tiempo de llenado y vaciado del pipeline.

Tiempo Ejec: Ni * CPI * Tciclo; CPI = 1 + Ciclos demora memoria + Ciclos demora saltos

DemMem = %accesosInst * %FallosInst * PenalizInst+ %accesosDatos * %FallosDatos * PenalizDatos
DemMem1 = 100% * 2% *(6 ciclos + 200ns/ 5ns/ciclo)+ (12%+8%)* 3% * (10 ciclos + 200ns/ 5ns/ciclo)=
= 1 * 0,02 *(46 ciclos)+ 0,2 * 0,03 * (50 ciclos)=
= 0,02*46 + 0,2*0,03*50 = 1,22 ciclos
DemMem2 = 100%* 2% * (6 ciclos + 200ns/ 4ns/ciclo)+ (12%+8%)* 3% *(10 ciclos + 200ns/ 4ns/ciclo)=
= 1 * 0,02 * (56 ciclos)+ 0,02 * 0,03 *(60 ciclos)=
= 0,02*56 + 0,2*0,03*60 = 1,48 ciclos

DemSaltos = %Salto_Incond * DemIncond + %Salto_cond*(%ef * demEf + %Nef*demNef)
DemSaltos1 = 2% * 1 + 10%*(33%ef *2 + 66%*0) = 0,02 + 0,1*(0,66+0) = 0,09
DemSaltos2 = 2% * 2 + 10%*(33%ef *3 + 66%*0) = 0,04 + 0,1*(0,99+0) = 0,14

CPI_1 = 1 + 1,22 + 0,09 = 2,31 ciclos
CPI_2 = 1 + 1,48 + 0,14 = 2,62 ciclos
T_CPU_1 = NI * 2,3 ciclos * 5 ns/ciclo = 11,55 ns * NI
T_CPU_2 = NI * 2,61 ciclos * 4 ns/ciclo = 10,48 ns * NI

CPU2 es más rápido que CPU1.
T_CPU_1 / T_CPU_2 = (11,55 ns * NI) / (10,48 ns * NI) = 1,1021 => 10,21 %

b. Al segundo sistema se le rediseña el acceso a memoria principal agregando un segundo nivel de caché (L2). Con esto, las tasas de fallos locales en la caché de instrucciones son fi1 = 2% y fi2=5%, y para caché de datos fd1=3% y fd2=10% siendo la penalización por fallo en caché de 10 y 60 ciclos respectivamente (TB1 = 10 ciclos y TB2=60). ¿En qué porcentaje mejora al sistema PROC_2? Responda en el sitio correspondiente.

Los tiempos de acierto en caché de primer nivel no penaliza. Es decir está contemplado en el CPI = 1.
DemMem3 = DemInstr + DemDatos
DemInstr = %accesos * (t_acceso1 + Fi1*(t_acceso2 + Fi2*Dem)))
= 100%*(0 + 2%*(10 + 5%*60)) = 1*(0,02*(10+0,05*60)) = 0,26 ciclos
DemDatos = %accesos * (t_acceso1 + Fi1*(t_acceso2 + Fi2*Dem)))
= (12%+8%) * (0 + 3%*(10 + 10%*60)) = 0,2*0,03*(10+0,1*60) = 0,096 ciclos
DemMem3 = 0,26 + 0,096 = 0,356 ciclos

T_CPU_3 = NI * CPI3 * Tciclo;
CPI3 = 1 + DemMem3 + DemSaltos
CPI3 = 1 +0,356 + 0,14 = 1,496 ciclos
T_CPU_3 = NI *1, 496 ciclos * 4 ns/ciclo = 5,984 ns * NI
Mejora CPU_3/CPU2 = 10,48 ns * NI / 5,984 ns * NI = 1,7513 = 75,13%

Usando Amdhal
Fm, es la parte del tiempo en acceso a memoria respecto del tiempo total:
Fm = 1,48 / (1 + 1,48 + 0,14) = 1,48 ciclos / 2,62 ciclos = 56,48%
Acc = 1,48 / 0,356 = 4,15
Acc = 1/((1-fm) + fm/acc)) = 1/(43,52% + 56,48%/4,15) = 1/0,5709 = 1,7513 => 75,13%

3.2.1. Se dispone de un sistema de memoria virtual paginado en dos niveles, en el que para el acceso a memoria se necesitan 4 ciclos, 2 para la traducción de la dirección virtual-real y dos adicionales para el acceso a los datos. En la tabla adjunta se muestra la estadística de uso de un conjunto de instrucciones y los ciclos promedio para la ejecución de cada tipo. Del los CPI indicados, 4 ciclos corresponden a la fase de acceso a memoria señaladas anteriormente.

Tipo Instrucción	LOAD/STORE	Saltos	Coma Flotante	Enteros
Promedio de uso	15%	10%	30%	45%
CPI	8	6	8	5

Se plantean dos mejoras independientes, la primera consiste en añadir una unidad específica para las instrucciones de coma flotante que implica reducir el tiempo de ejecución a la mitad. La segunda mejora pretende optimizar el acceso a memoria. Para ello se incorpora un TLB y una caché real. Las características de estas dos nuevas unidades son: tiempo de acceso al TLB 1 ciclo y frecuencia de fallos en el mismo del 5%. El acceso a la caché (1 palabra por bloque) es también de 1 ciclo y la frecuencia de fallos del 8%. Cada fallo se penaliza como si no hubiera habido mejora. Se quiere saber utilizando necesariamente la ley de Amdahl:
a) La mejora obtenida en ambos casos. ¿En cuánto es mejor una que la otra?
b) ¿Cuál será el rendimiento de la mejora conjunta del sistema?

SOLUCION:

a) $CPI^{ANTES} = 0,15*8 + 0,1*6 + 0,30*8 + 0,45*5 = 1,2 + 0,6 + 2,4 + 2,25 = 6,45$ ciclos

a1) La mejora de la UCF supone una mejora de $A_m = 2$ y se aplica en

$F_m = 0,3*4 / 6,45 = 1,2 / 6,45 = 0,186$

$A_G = 1 / \{(0,814) + (0,186/2)\} = 1,10.$

a2) La mejora realizada implica una mejora promedio que viene definida por la expresión:

$/t_{acc} =(t_{TLB} + F_{TLB} * Pen_{TLB})+ (t_{CACHÉ} + F_{CACHÉ} * Pen_{CACHÉ}) = (1 + 0,05*2) + (1 + 0,08*2) = 2,26$ ciclos.

Por lo tanto la aceleración mejorada será $A_m = 4 / 2,26 = 1,77$.

La mejora en el sistema de memoria afecta a la captura de instrucciones (1) y a las instrucciones con acceso para datos (0,15).

$F_m = (1+0,15)*4 / 6,45 = 0,713$

$A_G = 1 / \{(0,287) + (0,713 / 1,77)\} = 1,45$

La segunda mejora es 1,45 / 1,10 = 1,32 veces mejor que la primera (32% mejor)

b) Para tratar ambas mejoras al tiempo partimos de la segunda ya realizada y se aplica la primera sobre los resultados ya mejorados.

$CPI^{MEJORADO} = 0,15*(2,26+2,26) + 0,1*(2,26+2) + 0,30*(2,26+4) + 0,45*(2,26+1) = 0,678 + 0,426 + 1,878 + 1,467 = 4,45$ ciclos.

La mejora de la UCF supone una mejora de $A_m = 2$ y se aplica en:

$F_m = 0,30*4 / 4,45 = 0,270$

$A_G = 1 / \{(0,73) + (0,270/2)\} = 1,16.$

La mejora global será de 1,45 * 1,16 = 1,682. Es decir de 68%.

También se puede hacer con Amdahl generalizada: $A_G = 1 / \{1 – 0,186 – 0,713 + (0,186/2) + (0,713/1,77)\} = 1,68.$

3.2.2. Suponer que un sistema de memoria virtual tiene páginas de 4kbytes, dispone de un TLB completamente asociativo de 4 entradas como el de la figura. Se pide: **a)** Indicar el tamaño en bytes de la memoria virtual y real y **b)** Convertir la dirección virtual 1234567816 en la dirección real correspondiente.

P.V.	M.P.R.
0FF78	F27
01234	C43
12345	2F0
A5678	0B5

SOLUCION:

a) Como el offset es de 12 bits, en el TLB se encuentran la página virtual y el marco de página real respectivamente, por ello la memoria virtual será de 4 Gbytes y la real de 16 Mbytes.

b) Accediendo con la etiqueta 12345 a la tercera entrada se obtiene el marco 2F0, por ello la dirección real será $2F0678_{16}$

3.2.3. Un determinado sistema permite el acceso virtual a 4 Gbytes de memoria y dispone de un mapa de memoria física de 1 Mbyte. El sistema virtual es paginado puro con tamaño de página de 4 kbytes. Para la traducción de una dirección virtual (DV) a una real (DR), dispone de una unidad TLB, común para código y datos, asociativa de 2 vías, que guarda un total de 32 direcciones virtuales. Se pide:
a) Los campos en los que se divide la DV para el acceso al TLB. Indicar la estructura y tamaño del TLB. Suponer que no existe ningún bit de control.
b) Suponiendo el TLB lleno, indicar razonando la respuesta, el número máximo de referencias a memoria que puede generar la CPU sin que se produzca un fallo de página en el TLB.
c) Suponiendo el TLB vacío, señalar el estado del mismo tras la ejecución del siguiente programa, suponiendo que el bucle se ejecuta cinco veces. En cada caso elegir cualquier dirección real válida.

Dirección	**Instrucción**	**Comentario**
00FFF000	CLR A1	; Limpia el contenido del registro A1
00FFF002	L1: LOAD D1, $30H(A1)	; Lee memoria con direccionamiento relativo a registro
00FFF004	ADD D2, D1	; D1+ D2= D2
00FFF006	INC A1	; A1+ 1 = A1
00FFF008	INC A1	; A1+ 1 = A1
00FFF00A	JMP L1	; Salta a la etiqueta L1

NOTA: suponga que la DV: 00FFF000 se ha mapeado en la DR:32000
y que la DV: 00000000 se ha mapeado en la DR: 20000

SOLUCION:

a) Una memoria virtual de 4GB supone DV de 32 bits. Una memoria real de 1MB supone DR de 20 bits. El tamaño de página de 4kB implica reservar 12 bits para el offset.

Un TLB de dos vías con un total de 32 páginas supone 16 páginas por vía. Implica reservar 4 bits para el índice. Por tanto la DV se divide en:

ETIQUETA	INDICE	OFFSET
16 bits	4 bits	12 bits

El tamaño del TLB será 16 x 2 x (16 + 8) = 768 bits = 96 Bytes.

b) Cada entrada al TLB da acceso a una página con 2^{12} direcciones. Por tanto el número total de referencias a memoria diferentes será de 32 x 2^{12} = 2^{17} direcciones de memoria.

c) Todas las referencias a código, se encuentran en la misma página virtual la $00FFF_{16}$. Todas las referencias a datos se encuentran en la misma página virtual la 00000_{16}. Por tanto en el TLB sólo habrá dos entradas ocupadas de la primera vía con el índice F para las direcciones de código y con el índice 0 para las de datos. Es decir:

Indice	Etiqueta	Marco Página	Etiqueta	Marco Página
0	0000	32	- - -	- - -
F	00FF	20	- - -	- - -

3.2.4. Sea un sistema de memoria virtual paginado de 256Mbytes, con tamaño de página de 1kbyte. El sistema dispone de un TLB asociativo de 2 vías y un sistema multipaginado subdividido en niveles con tablas de páginas a su vez también paginadas que se encuentra en memoria principal. Calcular los campos en los que se divide la dirección virtual para el acceso a las tablas de páginas si se conoce que el tamaño del descriptor de página para cualquier nivel es de 16 bytes.

SOLUCION:

Si la DV tiene 28 bits y el OFFSET es de 10 bits, el campo marco de página virtual tiene 18 bits.

1.- Se supone un sistema de páginas subdividido en niveles (sistema multipaginado)

Si el tamaño de página es 210 bytes y el descriptor de página ocupa 2^4 bytes, implica que cada página tendrá 2^6 entradas. Es decir una tabla de páginas de acceso en cada nivel contendrá 64 páginas distintas, luego se necesitaran 6 bits para indexar dichas páginas. Por tanto el sistema de páginas está subdividido en tres niveles de 64 páginas cada uno. Por ello los campos en los que se divide la dirección virtual serán:

1er Nivel	2º Nivel	3er Nivel	OFFSET
6 bits	6 bits	6 bits	10 bits

3.2.5. Un determinado sistema de memoria dispone de un TLB completamente asociativo con 16 entradas con tamaño de página de 256 bytes. El sistema dispone de una unidad caché de 1 Kbytes asociativa de 4 vías con dos palabras por bloque. El sistema soporta 262.144 (256k) direcciones virtuales. El bus de direcciones y el de datos son ambos de 16 bits. Se pide:
a) El tamaño del TLB y de la caché cuando la caché es real. Explicar como se obtiene el resultado.
b) El tamaño del TLB y de la caché cuando la caché es virtual. Explicar como se obtiene el resultado.
NOTA: Considerar en ambos casos que no existen bits de control

SOLUCION:

Una memoria virtual de 256 kbytes supone DV de 18 bits. El bus de direcciones indica que las DR son de 16 bits. El tamaño de página de 256 bytes implica reservar 8 bits para el offset.

El TLB completamente asociativo de 16 entradas, debe guardar la página virtual (10 bits) y el marco de página real (8 bits).

Una caché de 1 kByte en 4 vías supone 256 Bytes/vía. Un bloque contiene 2 palabras cada una de 16 bits. Es decir hay 4 Bytes/bloque de un total de 64 bloques por cada vía.

a) En el caso de una caché real, se accede al TLB con la DV y a la caché con la DR.

Dirección Virtual	
ETIQUETA	OFFSET
10 bits	8 bits

El tamaño del TLB será 16 x (10 + 8) = 288 bits = 36 Bytes.

Dirección Real		
ETIQUETA	INDICE	OFFSET
8 bits	6 bits	2 bits

El tamaño de la caché será 64 (entradas) x 4 (vías) x {8 (etiqueta) + 32 (datos)} = 10.240 bits = 1280 Bytes.

b) En el caso de una caché virtual, se accede a la caché y al TLB siempre con la DV.

Dirección Virtual		
ETIQUETA	INDICE	OFFSET
10 bits	6 bits	2 bits

El tamaño de la caché será 64 (entradas) x 4 (vías) x {10 (etiqueta) + 32 (datos)} = 10.752 bits = 1344 Bytes.

El tamaño del TLB no cambia respecto al caso anterior.

3.2.6. En un Pentium se ha desactivado la segmentación y seleccionado un tamaño de página de 4 Kbytes. Su sistema de paginación consta de dos niveles. El primer nivel denominado directorio de páginas tiene 1024 entradas y la dirección base del directorio es la indicada por el registro CR3.
Suponiendo que el contenido en hexadecimal del registro **CR3** es **00033000** y el de algunas posiciones de memoria:

DIRECCION	DATO	DIRECCION	DATO	DIRECCION	DATO
00033000	03307000	**0021729C**	31320100	**003BC618**	00218063
00033700	003BC000	**0021829C**	535309AA	**003FF93C**	00113000
0011329C	67845260	**00337186**	003FF6A0	**03307186**	00217000

a) ¿De cuantas entradas dispone el segundo nivel (tabla de páginas)? **b)** Fijada una entrada del primer nivel (directorio de páginas) ¿Cuánta memoria se puede referenciar a partir de ella? Suponiendo que se quiere convertir la dirección virtual **7018629C**, se pide determinar: **c)** La dirección de memoria correspondiente para la entrada en el directorio de páginas. **d)** La dirección de memoria donde comienza la tabla de páginas. **e)** La dirección de memoria correspondiente para la entrada en la tabla de páginas. **f)** La dirección de memoria del dato referenciado y el valor contenido.

SOLUCION:

a) Si la tabla de páginas es de 4 kBytes, se necesitan 12 bits para el OFFSET, si el primer nivel de páginas (directorio) tiene 1024 entradas, implica que se necesitan 10 bits para indexarlas, por tanto con 32 bits de dirección virtual, quedan otros 10 bits para el segundo nivel de páginas.

Directorio	2º nivel páginas	OFFSET
10 bits	10 bits	12 bits

b) Con 2^{10} páginas de 2º nivel cada una de 2^{12} bytes se pueden direccionar 2^{22} bytes = 4 Mbytes.

La dirección virtual $7018629C_{16}$, se divide en los campos:

0111 0000 00 = $1C0_{16}$ como bits del directorio

01 1000 0110 = 186_{16} como bits de 2º nivel

0010 1001 1100 = $29C_{16}$ como OFFSET

Si cada descriptor tiene 4 bytes, cada índice a una tabla de páginas de cualquier nivel se alinea de 4 en 4 bytes, por tanto la entrada al directorio se calcula a partir del registro CR3 y de los bits de índice del directorio.

$00033000_{16} + 1C0_{16} \times 4 = 00033700_{16}$. El contenido de esta dirección señala a la dirección de la base de la tabla de páginas del 2º nivel.

c) La base de la tabla de páginas es $003BC000_{16}$

d) La dirección se obtiene de $003BC000_{16} + 186_{16} \times 4 = 003BC618_{16}$

e) En el descriptor de la posición de memoria $003BC618_{16}$ se obtienen los 20 bits del marco de página real, en este caso los primeros 20 bits de 00218063_{16}, es decir 00218_{16}

f) La dirección de memoria referenciada se obtiene con este marco y el OFFSET, por tanto la dirección es: $0021829C_{16}$ y el dato contenido $535309AA_{16}$

3.2.7. Un sistema ordenador de 32 bits, puede direccionar hasta 64Mbytes de memoria real, dispone además de las siguientes características: Espacio de direcciones virtuales de 1Gbyte con un tamaño de página de 4 Kbytes. TLB unificado completamente asociativo con 40 entradas y un byte de control por entrada. Unidad caché real y unificada de 20 Kbytes de capacidad, es asociativa por conjuntos con 5 vías, cada una con un tamaño de bloque de 32 bytes. Algoritmo de reemplazamiento LRU. Se pide:
a) Indicar en un esquema los campos de una entrada del TLB, indicando su tamaño. Calcular el tamaño total del TLB.
b) Indicar en un esquema los campos de una entrada de la unidad caché, indicando su tamaño. Calcular el tamaño total de la caché. No existen bits de control excepto los mínimos necesarios para el algoritmo LRU
c) Suponga una dirección virtual cualquiera que se encuentra en el TLB, explique como se calcula la correspondiente supuesta dirección real y con la ayuda de un esquema, indique como se accede a cada elemento de la caché, detallando que bits de la dirección se utilizan para extraer la palabra correcta

SOLUCION:

a) OFFSET = 12 bits (1 pag = 4 Kbytes)

Dirección Virtual	
Página Virtual	OFFSET
18 bits	12 bits

Dirección Real	
Marco Página	OFFSET
14	12 bits

La estructura de una entrada del TLB es:

Página Virtual	Control	Marco Página
18 bits	8 bits	14 bits

El tamaño del TLB será 40 x (18 + 8 + 14) = 1600 bits = 200 bytes

b) La estructura de una entrada de la CACHÉ REAL:

20kByte / 5 vías = 4kB/vía. 4 (kB/vía) / 32 (byte/bloque) = 128 bloques/vía, es decir 7 bits de índice.

La dirección real se subdivide para su acceso a la unidad caché en:

ETIQUETA	INDICE	OFFSET
14 bits	7 bits	5 bits

Para distinguir 1 entre 5 vías se necesitan 3 bits para el algoritmo LRU. La estructura de una entrada es:

VIA0		VIA1		VIA2		VIA3		VIA4		
DIR	DATOS	DIR	DATOS	DIR	DATOS	DIR	DATOS	DIR	DATOS	LRU
14 bit	32 By.	14 bit	32 By.	14 bit	32 By.	14 bit	32 By.	14 bit	32 By.	3 bits

Tamaño = 128entradas x (5 vías x (14b+32*8b) + 3b) = 173.184 bits = 21.648 Bytes

Hay que añadir cinco comparadores de 14 bits.

c) Por ejemplo, tomemos 3B2CF0AA como dirección virtual, (sólo los 30 lsb). El offset es 0AA y la página virtual viene indicada por 3B2CF (sólo 18 lsb bits).

1.- Se compara 3B2CF con las 40 etiquetas que contiene el TLB y suponemos que se produce un acierto en una cierta entrada. Sea el marco de página al que se accede 2456 (14 lsb bits). Al concatenar con el offset se obtiene la dirección real, por ejemplo 2456 0AA (los 26 lsb bits). Con esta DR dividida en tres campos se accede a la caché: Etiqueta: 10100010011100, índice: 0000101 y byte en el bloque: 01010.

2.- En la caché, los bits de índice señalan la sexta entrada, en esa entrada y para las cinco vías se compara la etiqueta. Si hay acierto se selecciona el bloque correspondiente, y dentro de este bloque se envía el dato direccionado por los cinco últimos bits.

3.2.8. Se tiene un sistema de memoria virtual de 64 Gbytes y con memoria física 16 veces menor. El sistema dispone de un TLB completamente asociativo con 32 entradas y una caché real asociativa de 4 vías y 256 kbytes de datos. Indicar, justificando brevemente la respuesta, a) el tamaño de página que permita el acceso en paralelo al TLB y a la unidad caché. b) Utilizando el valor anterior, calcular el tamaño en bits del TLB.

SOLUCION:

a) En un sistema de caché real, se accede al TLB utilizando los bits de la página virtual. Si los bits de OFFSET definidos por el tamaño de página, permiten acceder a la caché real este acceso se realiza en paralelo con el anterior. Si la caché de 256 kbytes es asociativa de 4 vías (64kbytes/vía), el tamaño de página que permite dicho acceso es de 64 kbytes (256 kbytes / 4 vías).

b) Las Direcciones Virtual y Real tendrán la forma:

Dirección Virtual	
Página Virtual	OFFSET
20 bits	16 bits

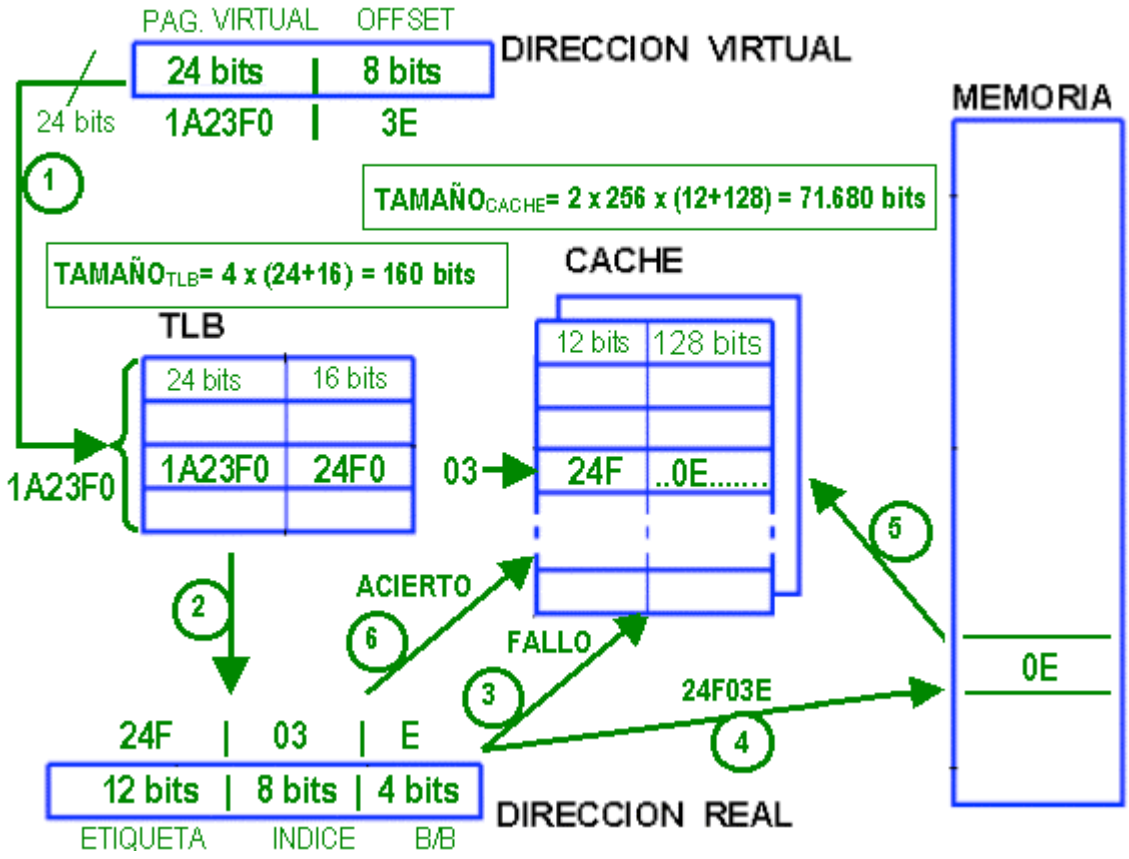
Dirección Real	
Marco Página	OFFSET
16	16 bits

El tamaño del TLB será: 32 x {20 (bits etiqueta) + 16 (bits marco) = 1152 bits}

3.2.9. Suponga que conoce los siguientes parámetros de un sistema: Dirección virtual (DV) de 32 bits. Dirección real (DR) de 24 bit. Paginación con tamaño de página de 256 Bytes. TLB completamente asociativo de 4 entradas. 8 KByte de caché real asociativa con 2 vías y guardando 16 bytes por bloque.

Indicar: **a)** Los campos en los que se dividirán la DV para trasladarla a DR y los campos de la DR para acceder a la caché o a memoria indicando su tamaño en bits. **b)** Tamaño del TLB y de la caché. **c)** Supóngase la caché vacía y que el procesador carga en un registro interno el contenido de la DV = $1A23F03E_{16}$. Esta DV se traslada en la DR = $24F03E_{16}$ mediante un acierto en el TLB y se sabe que el contenido de cada dirección de memoria coincide con el valor de los 4 bits menos significativos de la dirección (ejemplo: $F03423_{16}$ contiene 03). Se pide: **c1)** Rellenar el TLB, la caché y la memoria con los valores que corresponden al supuesto anterior, en la situación final, es decir, después de que el procesador haya cargado el registro. **c2)** Sobre el esquema por medio de flechas numeradas, indicar cómo y en qué orden se realizan los accesos a los distintos elementos del sistema.

SOLUCION:



3.2.10. Se tiene un sistema de memoria virtual con las siguientes características: Sistema paginado con un TLB completamente asociativo de 4 entradas y una caché real para datos de 8 Kbytes asociativa con 2 vías y guardando 8 bytes por bloque.
En un determinado momento, la CPU solicita leer un byte cuya dirección es $D35F072B_{16}$. Se produce un acierto en el TLB pero un fallo en la caché. El sistema accede a la dirección de memoria $45702B_{16}$ y se actualiza el bloque en la caché desde la cual la CPU puede leer el byte solicitado.
Con esta información se pide contestar, justificando brevemente la respuesta, las siguientes preguntas:
a) Tamaño en bits de las direcciones virtual y real:
b) Tamaño máximo de página:
c) Sin tener en cuenta ningún bit de control, el contenido de los campos de la entrada del TLB en donde se ha producido el acierto.
d) Sin tener en cuenta ningún bit de control, el contenido de todos los campos en la entrada de la caché donde se ha guardado el nuevo bloque. Se sabe que el contenido de las direcciones de la memoria real contiguas a la solicitada es:

Dirección (4570..)$_{16}$	..30	..2F	..2E	..2D	..2C	..2B	..2A	..29	..28	..27	..26	..25	..24
Byte guardado:	19	AA	23	45	76	89	F5	3D	66	3D	FA	2C	76

SOLUCION:

a) De las direcciones dadas, se obtiene que la dirección virtual ($D35F072B_{16}$) es de 32 bits y la real ($45702B_{16}$) de 24 bits

b) El tamaño de página viene dado por los bits del OFFSET iguales en ambas direcciones. Estos bits son los de menor peso que se repiten en ambas direcciones virtual y real. En este caso se repiten los 8 últimos ($2B_{16}$) por lo que el tamaño de la página será como máximo de 2^8 = 256 bytes.

c) En un TLB completamente asociativo, se guarda como mínimo la página virtual en el directorio y el marco de página real en el campo de datos, que en este caso eliminado el OFFSET en ambas direcciones se obtiene:

P.V.	**M.P.R.**
D35F07	**4570**

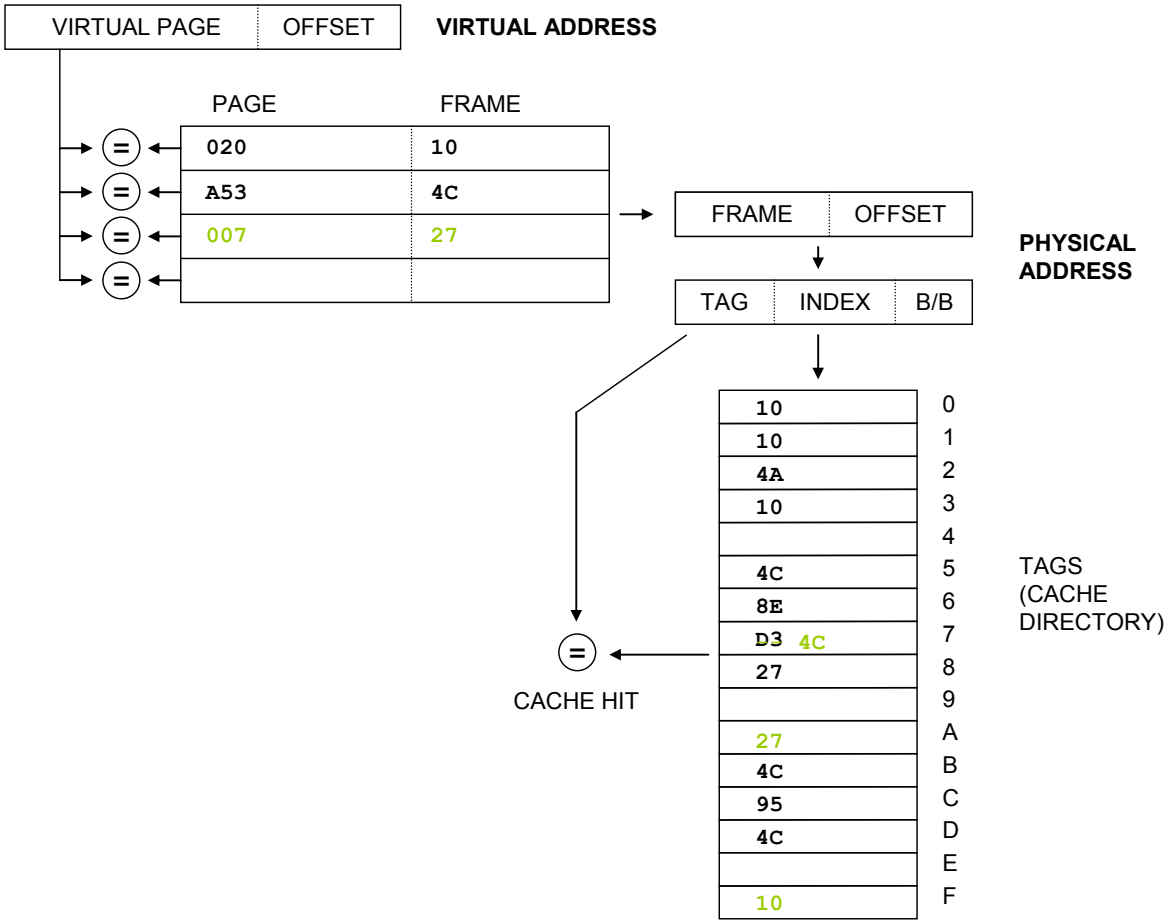
En una caché asociativa de 2 vías y con tamaño de 8 kbytes, cada vía contiene 2^9 = 512 entradas. Si cada bloque guarda 8 bytes la dirección debe reservar 3 bits para su identificación. Por tanto para el acceso a caché, la dirección real se divide en tres campos de tamaño y contenidos como los indicados:

ETIQUETA(12 bits)	**INDICE (9 bits)**	**BYTE/BLOQUE (3 bits)**
457_{16}	0000 0010 1_2	011_2

El byte señalado es el cuarto del bloque, por lo tanto, el contenido de los campos de la caché en la sexta entrada, la de índice 5, (0xx01012) de una de las dos vías debe ser:

DIRECTORIO (12 bits)	**BLOQUE (8 bytes)**
457_{16}	AA 23 45 76 89 F5 3D 66

3.2.11. La siguiente figura muestra un TLB junto a una caché (sólo aparece el directorio caché). Como se puede ver, el TLB tiene 4 entradas y la caché 16. El sistema de memoria usa paginación y tiene las siguientes



características: direccionamiento virtual de 1 MB, direccionamiento real o físico de 64 KB, páginas de 256 bytes. En este momento el TLB y la caché están parcialmente llenos, y se sabe que el rango de direcciones virtuales desde 0x00000 a 0x01FFF se corresponde con el rango de direcciones reales 0x2000 a 0x3FFF.

a) ¿Qué organización tiene el TLB? ¿Y la caché? ¿Es real o virtual? Justifica tus respuestas.

b) ¿Cuál es el tamaño de los campos PÁGINA VIRTUAL, OFFSET, MARCO, TAG, INDEX y B/B?

c) Suponiendo que la CPU envía la siguiente secuencia de direcciones (en hexadecimal): 02034 – 020F1 – A5354 – 007AB – 007A3 – 02015 – 0203A – A53B9 – A5377 – 02001. ¿Cuáles provocarán un fallo de página? ¿Y cuáles un fallo en la caché? Escriba sobre la figura los contenidos finales del TLB y del directorio caché, tachando los actuales si tienen que ser reemplazados.

SOLUCION:

a) El TLB tiene un comparador por cada entrada, lo que indica que no hay ninguna restricción en la posición que debe ocupar una determinada página virtual (es completamente asociativo). No es asociativo por vías (esto incluye a correspondencia directa) porque no hay bits de índice.

La caché es de correspondencia directa porque sólo hay un comparador, o sea, que una dirección cualquiera sólo se puede cachear en la entrada indicada por sus bits de índice. No es completamente asociativa porque tiene bits de índice, y tampoco es asociativa por vías porque en la figura no aparecen varias vías.

La caché es real porque trabaja con direcciones reales. Según se ve en la figura, las direcciones virtuales enviadas por la CPU pasan por el TLB antes de llegar a la caché. Por lo tanto a la caché le llegan direcciones reales.

b) Si el espacio de direccionamiento virtual es de 1 MB (20 bits) y las páginas tienen 256 bytes (8 bits), el tamaño del campo PÁGINA VIRTUAL será de 20-8=12 bits, y el del campo OFFSET de 8 bits.

Si el espacio de direccionamiento real es de 64 KB (16 bits) y el OFFSET contiene 8 bits, el campo MARCO ocupa 16-8=8 bits.

El campo INDEX ocupa 4 bits porque la caché tiene 16 bloques. Como se ve en la figura, el campo TLB tiene 8 bits. Por lo tanto, el campo B/B será de 16-4-8=4 bits. Por otro lado, el tamaño más adecuado (para optimizar su tiempo de acceso) de una caché real de correspondencia directa es una página. Si las páginas tienen 256 bytes, y la caché tiene 16 vías, el tamaño más adecuado para sus entradas es de 16 bytes, valor que concuerda con los 4 bits necesarios para el campo B/B calculados anteriormente.

c) **02034** ..... Acierto TLB y caché
A5354..... Acierto TLB y caché
007A3..... Acierto TLB y caché
0203A..... Acierto TLB y caché
A5377..... Fallo caché
020F1.....Fallo caché
007AB.....Fallo de página y de caché
02015.....Acierto TLB y caché
A53B9.....Acierto TLB y caché
02001.....Acierto TLB y caché

3.2.12. Se dispone de un ordenador con un procesador cuyas direcciones virtuales son de 24 bits y que dispone de una memoria real de 1 Mbyte. Se opta por un sistema de memoria paginado, con un tamaño de página de 256 bytes y longitud del descriptor de página de 2 bytes, en el que se incluye el marco de página, un bit de presencia y otros bits de control. Como la tabla de páginas es muy grande, ésta a su vez está paginada en tres niveles y sólo se guarda una parte en memoria real (los dos últimos niveles). La unidad de manejo de memoria (MMU) del sistema dispone de una unidad TLB completamente asociativa de 8 entradas. Además del TLB, dispone de una memoria de sustitución directa, que contiene todos los descriptores del primer nivel de las páginas en las que se ha dividido la tabla de páginas. El tamaño de la ruta de datos y por tanto el tamaño de las lecturas es de 16 bits. Cada subtabla de la tabla de páginas multinivel ocupa 1 página. Se pide:

a) El tamaño del TLB y de la memoria de sustitución directa presente en la MMU, así como la longitud y campos de los descriptores de página en ella guardados.

b) Indicar los pasos que debe seguir la MMU cuando el procesador solicite las siguientes lecturas: $EF2C58_{16}$ y $CC25B4_{16}$ y qué resultado se devolverá al procesador.

c) ¿Cuántas excepciones por falta de página se pueden producir como máximo en un acceso a memoria en este sistema? Justificar brevemente.

El tamaño de la ruta de datos y por tanto el tamaño de las lecturas es de 16 bits. En la figura se representa el contenido de la MMU y de algunas posiciones de memoria relevantes para el problema. Los datos están organizados “big endian” (byte de mayor peso en dirección más baja). La información del marco está en los bits de mayor peso de las palabras. Suponer que la página que se busca está presente.

Contenido del TLB

Etiqueta	M.P.R.	Control
2345	CCE	X
F356	011	X
EF2C	E0E	X
0001	F2E	X
112D	001	X
0303	123	X
3454	2D4	X
ABC1	DED	X

Memoria Sustitución Directa
(Directorio del Nivel 1)

Posición	M.P.R.	Control
0	CCC	X
1	321	X
2	132	X
3	3B2	X

MMU

Contenido de la Memoria Principal

Pos(hex)	Val(H)	Pos(hex)	Val(H)
00000	XX	1B001	A2
…	…	1B002	AA
00100	12	1B003	22
00101	13	…	…
00102	14	3B22E	31
00103	24	3B22F	32
…	…	3B230	55
01130	22	3B231	CF
01131	23	3B232	32
01132	24	…	…
01133	25	55400	22
…	…	55401	23
132D4	E0	55402	24
132D5	E3	…	…
…	…	55C48	CC
1B000	A0	55C49	CC

Pos(hex)	Val(H)	Pos(hex)	Val(H)
55C4A	CC	CCDB6	56
55C4B	DD	CCDB7	78
…	…	…	…
8010F	45	DED09	40
80110	67	DED0A	20
80111	88	DED0B	06
80112	98	…	…
…	…	E0E58	22
CCD57	01	E0E59	DE
CCD58	33	E0E5A	D3
CCD59	32	…	…
CCD60	56	F1A0B	BB
CCD61	78	F1A0C	DE
…	…	…	…
CCDB4	20	FFFFE	XX
CCDB5	05	FFFFF	XX

SOLUCION:

a)

Campos de la Dirección Virtual

Nivel 1	Nivel 2	Nivel 3	Offset
2	7 bits	7 bits	8 bits

Campos del TLB

Etiqueta	MPR	Control
16 bits	12 bits	4 bits

Dirección Real

M.P.R	Offset
12 bits	8 bits

Tamaño de campos del descriptor

M.P.R	control
12 bits	4 bits

DV: 24 bits; DR: 20 bits; Páginas: 256 Bytes = 2^8 Bytes => 8 bits de offset

En las tablas de páginas alojadas en páginas se pueden guardar 256/2 = 128 = 2^7 descriptores

Si hay 2 niveles de tabla en memoria queda para el primer nivel:

24(total)-8(offset)-7(nivel3)-7(nivel2) = 2 bits para el nivel 1.

Tamaño Total TLB: 8 x 32 bits = 256 bits = 32 Bytes

Tamaño Total primer nivel en MMU:

La tabla de sustitución directa posee 2^2 = 4 posiciones de 16 bits = 64 bits = 8 bytes

b) Indicar los pasos que debe seguir la MMU cuando el procesador solicite las siguientes lecturas: $EF2C58_{16}$ y $CC25B4_{16}$ y que resultado se devolverá al procesador.

b.1) Dir Virtual: $EF2C58_{16}$; Dir Real: $E0E58_{16}$; Dato Obtenido: $22DE_{16}$

Dada la dirección: $EF2C58_{16}$; Página Virtual: $EF2C_{16}$; Offset: 58_{16}.

La página virtual $EF2C_{16}$ se encuentra en el TLB. Luego la DR = $E0E58_{16}$. El contenido de memoria es $22DE_{16}$

b.2) Dir Virtual: $CC25B4_{16}$; Dir Real: $CCDB4_{16}$; Dato Obtenido: 2005_{16}

Dada la dirección: $CC25B4_{16}$; Página Virtual: $CC25_{16}$; Offset: $B4_{16}$. La P.V. no está en el TLB.

1100 1100 0010 0101 => L1=11 (3_{16}); L2 = 0011000 (18_{16}); L3 = 0100101 (25_{16});

En la memoria de sustitución directa el índice 3 devuelve un marco $3B2_{16}$.

La dirección en la tabla de nivel 2 será: $3B200_{16}$ + 18_{16}*2 = $3B230_{16}$. La entrada de la tabla es el valor $55CF_{16}$. El marco de la tabla de nivel 2 es $55C_{16}$.

La entrada en la tabla de nivel 3 es: $55C00_{16}$ + 25_{16}*2 = $55C4A_{16}$. El contenido de ésta entrada es: $CCDD_{16}$. Luego el marco es CCD_{16}.

La dirección real es DR: CCDB4. El contenido 2005_{16}.

c) El máximo de ocurrencias de fallos es tres. Si la dirección no está en el TLB no se produce excepción. Sólo cuando en el descriptor de páginas el bit de presencia está a 0 se produce una excepción para traer la página de Nivel 2. Si al acceder a ésta el bit de presencia nuevamente está a cero se traerá la página que contiene la tabla de páginas de nivel 3. Por último, al componer la DR puede que no esté en memoria y se producirá el tercer y último fallo de página.

3.2.13. Se dispone de un ordenador con un procesador cuyas direcciones virtuales son de 48 bits y que dispone de una memoria real de 4 Gbyte. Se opta por un sistema de memoria paginado, con un tamaño de página de 64 Kbytes y longitud del descriptor de página de 4 bytes en el que se incluye, el marco de página, un bit de presencia y varios bits de control más. Como la tabla de páginas es muy grande, ésta a su vez está paginada en tres niveles y sólo se guarda una parte en memoria real (los dos últimos niveles). La unidad de manejo de memoria (MMU) del sistema dispone de una unidad TLB 4-asociativo de 128 posiciones. Además del TLB, dispone de una memoria de sustitución directa, que contiene todos los descriptores del primer nivel de las páginas en las que se ha dividido la tabla de páginas. Se pide, justificando la respuesta, señalar:
a) Campos en que se divide la dirección virtual para el acceso al TLB y para el acceso a las tablas, la dirección real, los campos del TLB y los descriptores de página
b) El tamaño del TLB y de la memoria de sustitución directa presente en la MMU
c) El tamaño total de las tablas de páginas (solo nivel 2 y nivel 3). Si todas las páginas estuviesen en memoria que porcentaje representan respecto del total de memoria.
d) Cuantos accesos a memoria principal se deben realizar para obtener un dato, si la dirección no está en el TLB

SOLUCION:

a) DV: 48 bits; DR: 32 bits; Página: 64 KBytes = 2^{16} Bytes => 16 bits de offset

En una tabla de páginas, del tamaño de una página, se pueden guardar $2^{16}/ 2^2 = 2^{14}$ descriptores

Si hay 2 niveles de tabla en memoria queda para el primer nivel:

48 (DV) - 16(OFFSET) - 14(N3) - 14(N2) = 4 bits para el nivel 1 (N1).

Campos de la Dir.Virtual acc pag(48bits)

Pag. Virtual 32 bits

Nivel 1	Nivel 2	Nivel 3	Offset
4	14 bits	14 bits	16 bits

Campos del TLB (59 bits)

Etiqueta	MPR	Control
27 bits	16 bits	16 bits

Campos de la Dir.Virtual. acc TLB(48bits)

[Pag. Virtual 32 bits]

Etiqueta	Indice	Offset
27 bits	5 bits	16 bits

Dirección Real (32 bits)

M.P.R	Offset
16 bits	16 bits

Campo de tabla pág. (32 bits)

M.P.R	control
16 bits	16 bits

b) Tamaño del TLB: 128 x 59 bits = 7552 bits = 944 Bytes

La tabla de sustitución directa posee 2^4 = 16 posiciones de 32 bits = 512 bits = 64 bytes.

c) Tablas del N2: $2^4 \times 2^{14} \times 2^2$ = 16 tablas de 16.384 entradas de 4 Bytes = 2^{20} Bytes

Tablas del N3: $2^{18} \times 2^{14} \times 2^2$ = 262.144 tablas de 16.384 entradas de 4 Bytes = 2^{34} Bytes

Total = $2^{34} + 2^{20}$ = 16 GB + 1 MB = 16.385 MB

Representa aproximadamente el $2^{34} / 2^{32} * 100$ = 400% Es decir supera en un factor 4 la capacidad de la memoria real.

d) Se necesita un mínimo de 3 accesos a memoria principal:

La tabla de nivel 1 está presente en la MMU y no representa acceso a memoria. Luego un acceso a tabla de nivel 2, otro a la tabla del nivel 3 y finalmente el acceso al dato.

3.2.14. Se dispone de un sistema de memoria virtual de 16 MBytes, con una memoria real de hasta 1 MByte. El sistema es paginado con tamaño de página de 4 kBytes y cuenta con los siguientes elementos: caché virtual unificada, asociativa de 4 vías, con capacidad para guardar un total de 64 bloques de 32 bytes cada uno. En la MMU se dispone de un TLB completamente asociativo con 8 entradas. Se muestran algunas de las entradas en el estado actual de la caché, el TLB y parte de la memoria real. Se pide, tras ejecutar la secuencia de referencias a memoria siguiente: 4F20AE -- F50D80 – AA45F0 – F50D9C.

a) El tamaño total de la caché y del TLB, señalando en cada caso el número de bits de memoria que se necesitan y el número y tamaño de los comparadores. Para ello se ignorarán los posibles bits de control.

b) Teniendo en cuenta que los bloques LRU son los contenidos en la VIA 1 de la caché, se pide el estado final de la caché (señalar sólo aquellos casos en los que se cambia el contenido).

c) Indicar, justificando la respuesta, la estrategia elegida por la caché para las escrituras.

Nota: Para facilitar la lectura, en la etiqueta se utiliza una notación “mixta” con dígitos en hexadecimal (negrita) y en binario. Para los bloques sólo se indica a la izquierda el byte con la dirección de memoria más alta y a la derecha el de dirección de memoria más baja de los 32 bytes que contiene un bloque.

	VIA 1		VIA 2	VIA 3	VIA 4
0	**AA4** 000	**FF. 33**	- - - -	- - - -	- - - -
--	- - - - - -	- - - - - -	- - - -	- - - -	- - - -
--	- - - - - -	- - - - - -	- - - -	- - - -	- - - -
5	**4F2** 000	**4C. . . . .AA**	- - - -	- - - -	- - - -
--	- - - - - -	- - - - - -	- - - -	- - - -	- - - -
--	- - - - - -	- - - - - -	- - - -	- - - -	- - - -
C	**55F** 000	**22. . . . . .55**	- - - -	- - - -	- - - -
--	- - - - - -	- - - - - -	- - - -	- - - -	- - - -
--	- - - - - -	- - - - - -	- - - -	- - - -	- - - -
F	**100** 101	**D3. . . . . C4**	- - - -	- - - -	- - - -

TLB	
AA4	**CC**
4F2	**F0**
324	**D5**
55F	**35**
DF0	**C7**
F50	**50**
AAA	**10**
100	**2C**

MEMORIA									
DIRECC.	**DATO**	**DIRECC.**	**DATO**	**DIRECC.**	**DATO**	**DIRECC.**	**DATO**	**DIRECC.**	**DATO**
10020	**C4**	**35180**	**FF**	**50D80**	**44**	**CC5E0**	**66**	**F00A0**	**AA**
10021	**2F**	**35181**	**44**	**50D81**	**5C**	**CC5E1**	**42**	**F00A1**	**FC**
- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -
- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -
1003E	**00**	**3519E**	**33**	**50D9E**	**32**	**CC5FE**	**D3**	**F00BE**	**00**
1003F	**3F**	**3519F**	**22**	**50D9F**	**F4**	**CC5FF**	**F5**	**F00BF**	**4C**

a1) Tamaño total de la caché:

4 comparadores de 15 bits

4 vías x 16 entradas x (15 + 32*8) bits = 17344 bits de memoria = 2168 Bytes

a2) Tamaño del TLB:

8 comparadores de 12 bits

8 entradas x (12 + 8) bits = 160 bits de memoria = 20 Bytes

b) Las entradas modificadas de la caché son las siguientes.

	VIA 1		VIA 2	VIA 3	VIA 4
C	**F50** 110	**F4. . . . 44**	- - -	- - -	- - -
--	- - - -	- - - -	- - -	- - -	- - -
F	**AA4** 010	**F5. . . .66**	- - -	- - -	- - -

c) La estrategia empleada para escrituras es: post escritura, ya que no hay coherencia entre la caché y la memoria principal, tal como se observa en la dirección real 35180 (0xFF) que se encuentra almacenada como primer Byte del bloque en la entrada C de la caché (0x55).

3.2.15. Se tienen dos sistemas RISC diseñados con el mismo sistema de memoria, cuyas principales características son: Memoria virtual de 64Gbytes y real de 4 Gbytes. Tamaño de página 2 kbytes. Bus de datos de 32 bits. TLB con un total de 512 entradas repartidas en una estructura asociativa de 2 vías. Caché real unificada de 256 kbytes asociativa de 2 vías, con 8 palabras/bloque.
La diferencia principal entre ambas máquinas se basa en que la denominada MAQ1 dispone de un sistema de ventanas de registros tipo SPARC, mientras que la denominada MAQ2 no. La frecuencia de reloj en ambos sistemas es la misma, así como el porcentaje medio de instrucciones que junto con el coste en ciclos para cada tipo se muestran en la tabla adjunta:

Load/Store	Saltos	Llamadas a rutinas	Oper. ALU
33%	15%	2%	50%
1,2 ciclos	1,4 ciclos	1,4 ciclos	1,0 ciclos

Por tener distinto sistema de registros, la diferencia entre MAQ1 y MAQ2 viene dada las diferentes operaciones adicionales que se deben realizar para guardar y restaurar datos en las llamadas a rutinas y su posterior retorno.

* En la MAQ1, gracias al sistema de ventanas de registros, sólo se producen operaciones adicionales cuando se produce un *overflow* por la limitación en el número de ventanas del sistema. El *overflow* ocurre un 5% de las llamadas a rutina e implica una penalización de 50 instrucciones del tipo Load/Store, 10 Saltos y 20 operaciones con la ALU.
* En cada llamada a rutina, la MAQ2 necesita ejecutar un 7% adicional en operaciones del tipo Load/Store.

a) Indicar los campos en los que se divide la dirección virtual para el acceso al TLB y la dirección real para el acceso a la caché.
b) Suponiendo un sistema de memoria perfecto, se pide el CPI para cada máquina indicando cual de las dos es mas eficiente y en cuanto. Se ha analizado un determinado programa que utiliza un 40% de instrucciones con memoria. Se ha medido una tasa de fallos del 0,1% en el TLB y se quiere conocer el efecto de variar el tamaño del bloque de la caché. Se dispone de la siguiente información:

Tamaño Bloque (bytes)	16	32	64
Tasa de fallos	1,5%	1,0%	0,8%

Se sabe que la penalización por fallo en el TLB es de 25 ciclos y por fallo en la caché es de 6 ciclos más un ciclo adicional por cada palabra en el bloque. Se pide:
c) ¿Cuál es el tamaño óptimo de bloque que minimiza las pérdidas por accesos a memoria?
d) Para la MAQ1 la pérdida de rendimiento en el sistema al tener en cuenta el sistema real de memoria

SOLUCION:

a) Memoria virtual de 64 GB implica un DV de 36 bits. Si el tamaño de página es de 2 kbytes implica que el OFFSET debe ser de 11 bits.

Si el TLB es asociativo de 2 vías distribuye 256 entradas por vías, luego se necesitan 8 bits para indexar el TLB.

La Dirección Virtual para el acceso al TLB será de la forma:

ETIQUETA	INDICE	OFFSET
17 bits	8 bits	11 bits

Memoria real de 4 GB implica un DR de 32 bits. Si el tamaño del bloque es 8 palabras /bloque y el bus de datos de 32 bits, supone por tanto 32 bytes en un bloque y se necesitan 5 bits para su identificación.

Si la caché de 256 kbytes (2^{18}) es asociativa de 2 vías distribuye 128 kbytes (2^{17}) por vía, es decir dispone de 2^{12} entradas kbytes luego se necesitan 12 bits para indexar la caché.

La Dirección Real para el acceso a la caché será de la forma:

ETIQUETA	INDICE	Byte/Bloque
15 bits	12 bits	5 bits

b) Sin considerar la penalización por las llamadas a subrutinas el CPI de ambas máquinas debe ser igual a

$CPI = \Sigma\ CPI_i \times C_i = 0{,}33 \times 1{,}2 + 0{,}15 \times 1{,}4 + 0{,}02 \times 1{,}4 + 0{,}5 \times 1 = 1{,}134$ ciclos.

De forma adicional en cada sistema las operaciones de saltos generan la siguiente penalización:

MAQ1: El 5% de las llamadas se penaliza por *overflow* con un paquete adicional de instrucciones.

$CPI^P_{MAQ1} = 0{,}02 \times 0{,}05 \times \{50 \times 1{,}2 + 10 \times 1{,}4 + 20 \times 1\} = 0{,}094$ ciclos

$CPI_{MAQ1} = 1{,}134 + 0{,}094 = 1{,}228$ ciclos

MAQ2: El 100% de las llamadas se penaliza con un 7% adicional de instrucciones Load/Store.

$CPI^P_{MAQ2} = 0{,}02 \times 0{,}07 \times 1{,}2 = 0{,}002$ ciclos

$CPI_{MAQ2} = 1{,}134 + 0{,}002 = 1{,}136$ ciclos

La eficiencia del sistema MAQ2 es mayor en un 8% (1,228/1,136 = 1,08).

c) Sólo se deben considerar las penalizaciones en los accesos a memoria, tanto para instrucciones como para datos. Los aciertos se consideran dentro como CPI ideal. La penalización por fallo en el TLB es la misma para los tres sistemas y no tiene que contar en la comparación:

$CDM = \%A_{cc} \{(F_{TLB} \times P_{TLB}) + (F_{CACHE} \times P_{CACHE})\}$.

$CDM_{16B} = 1,4 \{(0,001x25) + (0,015 \times (6+4))\} = 0,035 + 0,210 = 0,245$ ciclos

$CDM_{32B} = 1,4 \{(0,001x25) + (0,01 \times (6+8))\} = 0,035 + 0,196 = 0,231$ ciclos

$CDM_{64B} = 1,4 \{(0,001x25) + (0,008 \times (6+16))\} = 0,035 + 0,246 = 0,281$ ciclos.

Por tanto el tamaño de bloque que minimiza las pérdidas por acceso a memoria es de 32 bytes.

d) Para MAQ1, debido a las rutinas de *overflow*, aumenta el número de accesos para captura de instrucciones respecto al caso ideal en un porcentaje dado por:

$\%A_{cc}^{I} = 0,02 \times 0,05 \times (50 + 10 + 20) = 0,08$.

Se debe considerar por tanto un porcentaje de 1,08 como acceso medio a memoria de instrucciones.

Además en MAQ1, el % de accesos a memoria de datos debe considerar el añadido debido a las operaciones de *overflow*, es decir:

$\%A_{cc}^{D} = 0,02 \times 0,05 \times 50 = 0,05$

Se trata por tanto de un 38% de instrucciones que acceden a datos.

$CDM_{MAQ1} = 1,46 \{(0,001x25) + (0,01 \times (6+8))\} = 0,241$ ciclos.

La pérdida de rendimiento del sistema MAQ1 respecto al sistema ideal de memoria es de ~16% (1,228/1,469 = 0,84)

3.2.16. Sea un sistema de memoria virtual, que opera con instrucciones y datos de 32 bits. El sistema dispone de una unidad TLB, completamente asociativa de 8 entradas y de una unidad caché virtual unificada, asociativa de dos vías, con 16 entradas por vía, 16 bytes por bloque y un algoritmo LRU para las sustituciones. Se facilitan algunos contenidos relevantes en el estado inicial del TLB, de la caché y de memoria. A continuación, se ejecuta el código adjunto:
NOTA: sólo en algunas ocasiones, el contenido facilitado de la caché es relevante.

```
PC          Instrucción              Comentario
0A040       MOV R4, 0                ; R4 <= 0
0A044       MOV R1, 2                ; R1 <= 2
0A048       MOV R2, A305Ch           ; R2 <= A305C16
0A04C  L1:  LOAD R3, R2, R0          ; R3 <= MEM[R2+0]
0A050       ADD R4, R4, R3           ; R4 <= R4 + R3
0A054       ADD R2, R2, 4            ; R2 <= R2 + 4
0A058       ADD R1, R1, -1           ; R1 <= R1 - 1
0A05C       BNZ L1                   ; Salta a L1 si Z = 0
0A060       JMP FIN                  ; Salta a FIN
- - - -
0A0E0  FIN: STORE R4, R2, A0h        ; R4 => MEM[R2 + A016]
0A0E4       HALT
```

TLB	
FC5	02
A30	B0
...	...
0BC	F5
0A0	3C

	UNIDAD CACHE			
	VIA 1		VIA 2	
0 =>	032	B0030200 A02A305C A0300002 A0400000	BC0	A0A2D4D5 88AA3460 45803D59 95203580
1 =>				
...	...	...	...	...
4 =>	0A0	0020300B C503A02A 2000001A 00000004A	3F2	A0A3D420 40FFDC60 458032FF 45203580
5 =>	4B2		A30	10102020 30405060 708090A0 B0C0D0E0
6 =>	A03		ABC	
...	...	...	...	...
E =>	0A0	10102020 30405060 708090A0 5060D0E0	3F0	A0A3D420 40FFDC60 458032FF 45203580
F =>	3FF	A0A3D420 40FFDC60 458032FF 45203580	4C0	

MEMORIA															
DIR	VAL	DIR	VAL	DIR	VAL	-	DIR	VAL	-	DIR	VAL	DIR	VAL	DIR	VAL
3C50	50	3C58	51	3C60	B0		A030	F4		B058	10	B060	40	B068	20
3C51	04	3C59	01	3C61	0F		A031	DE		B059	20	B061	40	B069	40
3C52	04	3C5A	01	3C62	FF		A032	C0		B05A	30	B062	A0	B06A	3F
3C53	03	3C5B	FF	3C63	80		A033	0A		B05B	40	B063	A0	B06B	20
3C54	51	3C5C	F0	3C64	3F		A034	45		B05C	50	B064	C0	B06C	4B
3C55	02	3C5D	0F	3C65	40		A035	74		B05D	60	B065	F5	B06D	BC
3C56	02	3C5E	FF	3C66	55		A036	22		B05E	70	B066	D6	B06E	D0
3C57	04	3C5F	F6	3C67	D0		A037	80		B05F	80	B067	FF	B06F	00

Con la información facilitada, se pide:
a) Justificando brevemente la respuesta, el tamaño de página así como el de la memoria virtual y real.

En el código del programa, se identifica que las direcciones (virtuales) son de 20 bits. En el TLB, el campo asociado a la página virtual es de 12 bits y el asociado al marco de página real 8, por tanto el offset es de 8 bits. Las respuestas correctas son

El tamaño de página es de 2^8 = 256 bytes

La dirección virtual tiene 20 bits, luego la memoria virtual es de 2^{20} = 1 Mbytes

La dirección real tiene 16 bits, luego la memoria real es de 2^{16} = 64 kbytes

b) Justificando brevemente la respuesta, el porcentaje de aciertos al ejecutar el código completo

Las tres primeras instrucciones se ejecutan una sola vez y están presentes en la caché: 0 Fallos, 3 Aciertos.

El bucle se ejecuta dos veces (5x2) y hay un fallo en la dirección 0A050 la primera vez: 1F, 9A

La instrucción 0A060, se ejecuta una sola vez y falla en el acceso: 1F, 0A.

Las dos últimas instrucciones se ejecutan una sola vez y están presentes en la caché: 0F, 2A

En el bucle, dos veces, se accede una vez a datos (LOAD), la primera vez (A305C) acierta, pero en la segunda (A3060) falla: 1F, 1A.

Fuera del bucle hay un único acceso a datos (STORE), la dirección (A3104) no se encuentra en caché: 1F, 0A.

En total, hay 4 Fallos y 15 Aciertos, la tasa de aciertos es 15/19 = 78,9%

c) Sabiendo que en la caché el byte más a la derecha del bloque se corresponde con la dirección más baja de memoria y que se utiliza notación "*big endian*", señalar en la tabla adjunta el valor en hexadecimal de los registros R2, R3 y R4, tras la ejecución del código.

RESULTADO	
Valor de R2:	$000A3064_{16}$
Valor de R3:	$4040A0A0_{16}$
Valor de R4:	$6060B0B0_{16}$

3.2.17. El ordenador educativo ARQUI01 dispone de un procesador cuyas direcciones virtuales son de 20 bits y dispone de una memoria real de 64 kBytes. Posee un sistema de memoria paginado, con un tamaño de página de 256 bytes y longitud del descriptor de página de 4 Bytes en el que se incluye el marco de página, bit de presencia y varios bits de control. Como la tabla de páginas es muy grande, ésta a su vez está paginada en dos niveles (el primero siempre está presente en la memoria principal). La unidad de manejo de memoria (MMU) del sistema dispone de un TLB completamente asociativo de 8 entradas. El tamaño de la ruta de datos y, por tanto, el tamaño de las lecturas es de 16 bits. El sistema posee además caché virtual unificada asociativa de 4 vías, 16 bloques en total y 2 palabras por bloque.
En las figuras se representa el contenido del TLB, el contenido de la caché y de algunas posiciones de memoria relevantes para el problema. Los datos están organizados "little endian" (byte de menor peso en dirección más baja). La información del marco está en los bits de menor peso de las 2 palabras. Suponer que la página que se busca está presente. Se pide:

a) Cómo se divide la dirección para el acceso a caché, TLB y memoria (rellenar las plantillas). Indicar el tamaño del TLB y de la caché. Exprese tanto el tamaño de datos como el tamaño total y cantidad de comparadores en el caso de la memoria caché.
b) Indicar los pasos que se han de seguir cuando el procesador genere las direcciones $E2C58_{16}$, $CC2BC_{16}$ y $1C2B2_{16}$ y qué resultado se devolverá al procesador.
c) Indique (en la tabla adjunta de la próxima página) en qué posiciones de la memoria caché habrá actualizaciones para los accesos a memoria del apartado b). Suponga que la vía LRU para todos los casos es la vía 4. No es necesario indicar en las tablas la actualización de la caché, sólo en qué posiciones.

Contenido del TLB

Etiq.	Ctrl	MPR
245	X…X	CE
F56	X…X	22
E2C	X…X	01
001	X…X	2E
11D	X…X	01
CC2	X…X	2B
344	X…X	D4
AB1	X…X	ED

Caché Virtual Unificada

	Vía 1		Vía 2		Vía 3		Vía 4	
	Etiq	Datos	Etiq	Datos	Etiq	Datos	Etiq	Datos
0	CCCC	ABCD EF12	CCCD	ABDC EF21	CDCC	ABCE EF13	CCFC	ABEC EF31
1	CCCC	ABCD EF14	CCCD	ABDC EF23	CDCC	ABCE EF15	CCFC	ABEC EF33
2	A132	2007 2006	E2C5	2008 2009	31B2	EABC 1ECD	1132	ABCD EF12
3	3B21	5E83 197F	1234	FA19 38B4	1C2B	29D3 8930	13B2	3830 5740

Base 1er nivel de páginas: 5500 hex

Contenido de la Memoria Principal

Pos(hex)	Val(H)
0000	XX
…	…
0100	12
0101	13
0102	14
0103	24
…	…
0158	09
0159	20
015A	08
015B	20
…	…
13D4	E0
13D5	E3
…	…
2BBC	A0

Pos(hex)	Val(H)
2BBD	A2
2BBE	AA
2BBF	22
…	…
3B2E	31
3B2F	32
3B30	55
3B31	CF
3B32	32
551C	CC
551D	22
551E	24
551F	DD
…	…
5C49	CC

Pos(hex)	Val(H)
5C4A	CC
5C4B	DD
…	…
810F	45
8110	67
8111	88
8112	98
…	…
C857	01
C858	33
…	…
CC08	E0
CC09	E1
…	…
CDB4	20
CDB5	05

Pos(hex)	Val(H)
CDB6	56
CDB7	78
…	…
DE09	40
DE0A	20
…	…
E0B0	21
E0B1	EE
E0B2	D0
E0B3	DE
…	…
F1AB	BB
F1AC	DE
…	…
FFFE	XX
FFFF	XX

ORGANIZACIÓN Y ESTRUCTURA DE LA MEMORIA: CACHÉS Y MEMORIA VIRTUAL

a)

Dir. virtual para acceso a tablas

Nivel 1	Nivel 2	Offset
6 bits	6 bits	8 bits

Campos del TLB

Etiqueta	Control	MPR
12 bits	24 bits	8 bits

Dirección real

MPR	Offset
8 bits	8 bits

MPR: Marco de Pág. Real

Tamaño de campos del descriptor

Control	MPR
24 bits	8 bits

Campos de la dirección virtual para acceso a la caché

Etiqueta	Índice	B/B
16 bits	2 bits	2 bits

Breve justificación y tamaños TLB y caché:

DV: 20 bits; DR: 16 bits; Páginas: 256 Bytes = 2^8 Bytes => 8 bits de offset

En las tablas de páginas alojadas en páginas se pueden guardar 256/4 = 64 = 2^6 descriptores

Si la tabla tiene 2 niveles: 20 (total) = 6 (nivel1) + 6 (nivel2) + 8 (offset)

Acceso a la caché 20 bits; 2 Byte en Bloque (2 palabras de 2 bytes cada una); 2 bits de índice ya que hay 4 bloques por vía (16 bloques/4 vías), Etiqueta: 20-2-2 = 16 bits

Tamaño total TLB: 8 x (12 bits etiquetas + 32 bits descriptor) = 352 bits = 44 Bytes

Tamaño caché, parte de datos: 4 vías x 4 bloques/vía x 2 palabras/bloque x 2 Bytes/pal = 64 Bytes

Tamaño total caché: 4 vías x 4 bloques/vía x (2 Bytes de etiqueta + 4 Bytes datos) = 96 Bytes

Comparadores caché: 4 comparadores de 16 bits

b) Indicar los pasos que se han de seguir cuando el procesador genere las direcciones $E2C58_{16}$, $CC2BC_{16}$ y $1C2B2_{16}$ y qué resultado se devolverá al procesador.

b.1) Dir Virtual: $E2C58_{16}$; Búsqueda en caché: Etiq: E2C5 con índice 10_2, acierto en vía 2; Dato Obtenido: **2009_{16}** ya que el B/B es 00 (parte baja).

b.2) Dir Virtual: $CC2BC_{16}$; Índice para acceso a caché = 11_2; Fallo en caché;

Acceso al TLB: Etiqueta: CC2; acierto en TLB, marco 2B; luego la DR=2BBC

Dato Obtenido: **$A2A0_{16}$** Direcciones (2BBD y 2BBC). Se carga en caché entrada 11_2 las posiciones de memoria 2BBF-2BBE-2BBD-2BBC (22AA A2A0) con etiqueta CC2B

b.3) Dir Virtual: 1C2B2; índice para acceso a caché: = 00_2; Fallo en caché;

Acceso al TLB: Etiqueta: 1C2; Fallo en TLB

Búsqueda en la tabla: #pág en binario 0001 1100 0010 => índice L1 =000111_2 = 7_{16}; índice L2=000010_2=2_{16};

Luego si la base de página es 5500, la entrada en la tabla de 1er nivel es: 5500_{16} + 7_{16}*4 = $551C_{16}$. Contenido DD2422CC. Es decir, el marco es CC.

El acceso al segundo nivel es $CC00_{16}$ + 2_{16}*4 = $CC08_{16}$. Siendo el marco buscado E0. Luego DR=E0B2.

Dato Obtenido**:** **$DED0_{16}$** Direcciones (E0B3 y E0B2). Se actualiza caché con las direcciones (E0B3, E0B2 E0B1 y E0B0). DED0 EE21, etiqueta 1C2B y TLB con etiqueta 1C2 y marco E0.

c) Indique (en la tabla adjunta a continuación) en qué posiciones de la memoria caché habrá actualizaciones para los accesos a memoria del apartado b). Suponga que la vía LRU para todos los casos es la vía 4. No es necesario indicar en las tablas la actualización de la caché, sólo en qué posiciones.

	Vía 1		Vía 2		Vía 3		Vía 4	
	Etiq	Datos	Etiq	Datos	Etiq	Datos	Etiq	Datos
0							1C2B	DED0 EE21
1								
2								
3							CC2B	22AA A2A0

b.1 No modifica caché.

b.2. Se carga en caché entrada 11_2 las posiciones de memoria 2BBF-2BBE-2BBD-2BBC (22AA A2A0), como la LRU es la vía 4 se coloca en ella. La etiqueta es CC2B.

b.3. Se actualiza caché con las direcciones (E0B3, E0B2, E0B1 y E0B0). DED0 EE21, etiqueta 1C2B.

3.2.18. Se dispone de un procesador con memoria virtual capaz de direccionar hasta 16 MBytes, mientras que la memoria física puede ser de hasta 1 MByte. El sistema es paginado puro con tamaño de página de 256 Bytes, siendo el tamaño de palabra de 2 Bytes. Se dispone de un TLB completamente asociativo de 8 entradas y de caché de dos niveles (L1 y L2), tratándose en ambos casos de caché real unificada. La caché de nivel 1 (L1) es asociativa de 4 vías, con 64 bloques por vía y tamaño de bloque 4 Bytes. Por su parte, la caché de nivel 2 (L2) es de correspondencia directa con 256 bloques de 16 Bytes cada uno.

Se muestra parcialmente el contenido del TLB, las dos cachés y la memoria real. Los contenidos no mostrados de las cachés se considerarán como fallos en caso de acceso.

Nota: para facilitar la lectura se utiliza una notación "mixta" con dígitos en hexadecimal (negrita) y en binario. Se representa siempre a la izquierda el byte más significativo y el sistema es *little endian* (byte menos significativo en la dirección más baja). No se muestran en ningún caso los bits de control.

	Caché L1				
	VÍA 1		VÍA 2	VÍA 3	VÍA 4
Índice	Etiqueta	Bloque			
000000	**C34**	**00010203**	**- - - -**	**- - - -**	**- - - -**
--	**- - - - - -**	**- - - - - -**	**- - - -**	**- - - -**	**- - - -**
000101	**A3F**	**8371AB04**	**- - - -**	**- - - -**	**- - - -**
--	**- - - - - -**	**- - - - - -**	**- - - -**	**- - - -**	**- - - -**
010101	**1C4**	**E739010C**	**- - - -**	**- - - -**	**- - - -**
--	**- - - - - -**	**- - - - - -**	**- - - -**	**- - - -**	**- - - -**
100001	**476**	**020D6593**	**- - - -**	**- - - -**	**- - - -**
--	**- - - - - -**	**- - - - - -**	**- - - -**	**- - - -**	**- - - -**
111111	**100**	**D3838FC4**	**- - - -**	**- - - -**	**- - - -**

TLB	
Nº pág.	Marco
AA40	**C30**
73BA	**2D5**
0302	**D58**
4F23	**1C4**
1DF0	**0B3**
F250	**A50**
C340	**3E6**
1040	**C36**

Caché L2

Índice	Etiqueta	Bloque
00	**C3**	**00010203 3764FC39 8371AC05 E739010D**
--	**- - - -**	**- - - - - -**
45	**1C**	**3764DC39 8371AB04 E739010C EAFC0384**
--	**- - - -**	**- - - - - -**
58	**2D**	**E739010C 02040608 4285A8F0 940D9438**
--	**- - - -**	**- - - - - -**
60	**C3**	**12983476 94E8C385 020D6593 00010203**
--	**- - - -**	**- - - - - -**
FF	**10**	**3764DD39 D3838EC4 3764DC3A E739020D**

MEMORIA									
DIRECC.	DATO	DIRECC.	DATO	DIRECC.	DATO	DIRECC.	DATO	DIRECC.	DATO
- - - -	**- -**	**- - - -**	**- -**	**- - - -**	**- -**	**- - - -**	**- -**	**- - - -**	**- -**
100FC	**C4**	**2D586**	**85**	**3E600**	**09**	**C3000**	**0D**	**C3600**	**03**
100FD	**8F**	**2D587**	**42**	**3E601**	**01**	**C3001**	**01**	**C3600**	**02**
- - - -	**- -**	**- - - -**	**- -**	**- - - -**	**- -**	**- - - -**	**- -**	**- - - -**	**- -**
1C454	**0C**	**34060**	**33**	**50D9E**	**32**	**C3008**	**39**	**A3F140**	**04**
1C455	**01**	**34061**	**22**	**50D9F**	**F4**	**C3009**	**FC**	**A3F141**	**AB**
- - - -	**- -**	**- - - -**	**- -**	**- - - -**	**- -**	**- - - -**	**- -**	**- - - -**	**- -**

a) Rellenar el número de bits utilizados para cada campo en los siguientes casos.

	Nº pág.	Offset
Direc. Virtual	**16**	**8**

	Marco	Offset
Direc. Real	**12**	**8**

	Etiqueta	Índice	B/B
Caché L1	**12**	**6**	**2**

	Etiqueta	Índice	B/B
Caché L2	**8**	**8**	**4**

b) Se utiliza una tabla de páginas en un solo nivel, con descriptores de 2 Bytes. ¿Cuánto ocuparía la tabla en caso de estar completa?

Hay un total de 2^{16} páginas virtuales distintas. Para cada página posible se necesitan 2 Bytes (descriptor), así que ocuparía un total de 2^{17} Bytes, es decir, 128 kBytes (una octava parte de la máxima memoria real disponible en este micro).

c) ¿Cuál sería el dato devuelto para cada una de las siguientes direcciones virtuales? **4F2354** – **73BA86** – **C34000**. Justifique brevemente la respuesta, indicando si hay éxito o fallo en TLB, L1 y L2.

- Dirección virtual **4F2354**. Acierto en TLB (**4F23** se traduce por **1C4**). Dirección real: **1C454**. Acceso a L1: etiqueta **1C4**, índice 010101 y B/B 00. Acierto en L1, palabra devuelta **010C**.
- Dirección virtual **73BA86**. Acierto en TLB (**73BA** se traduce por **2D5**). Dirección real: **2D586**. Acceso a L1: etiqueta **2D5**, índice 100001 y B/B 10. Fallo en L1. Acceso a L2: etiqueta **2D**, índice **58**, B/B **6**. Acierto en L2, palabra devuelta **4285**.
- Dirección virtual **C34000**. Acierto en TLB (**C340** se traduce por **3E6**). Dirección real: **3E600**. Acceso a L1: etiqueta **3E6**, índice 000000 y B/B 00. Fallo en L1. Acceso a L2: etiqueta **3E**, índice **60**, B/B **0**. Fallo en L2. Acceso a memoria con dirección **3E600**, palabra devuelta **0109**.

d) Suponiendo que el acceso a MMU se da siempre antes de empezar el acceso a cachés, se pide calcular el tiempo medio de acceso a memoria (en ciclos) con los siguientes datos. Tiempo de acceso a TLB t_{TLB}=1, tiempo de acceso a tablas t_{tablas}=15, tiempo de acceso a caché L1 t_{L1}=1, tiempo de acceso a caché L2 t_{L2}=3, tiempo de acceso a memoria t_{mem}=10. Los buses entre L1 y L2, y entre L2 y memoria son ambos de 4 Bytes de ancho, y el tiempo de acceso incluye acceder a todo el ancho de bus. La tasa de acierto local en L1 es del 90%, en L2 del 80% y en TLB del 98%, y se supone que nunca se falla en tablas de páginas.

Si se falla en L1, traer un bloque a L1 desde L2 requiere un único acceso (4 Bytes). Por tanto, $t_{B1}=t_{L2}$=3. Pero si se falla en L2, traer un bloque a L2 desde memoria requiere 4 accesos (16 Bytes), por lo que esta penalización es de $t_{B2}=4\cdot t_{mem}$=40 ciclos.

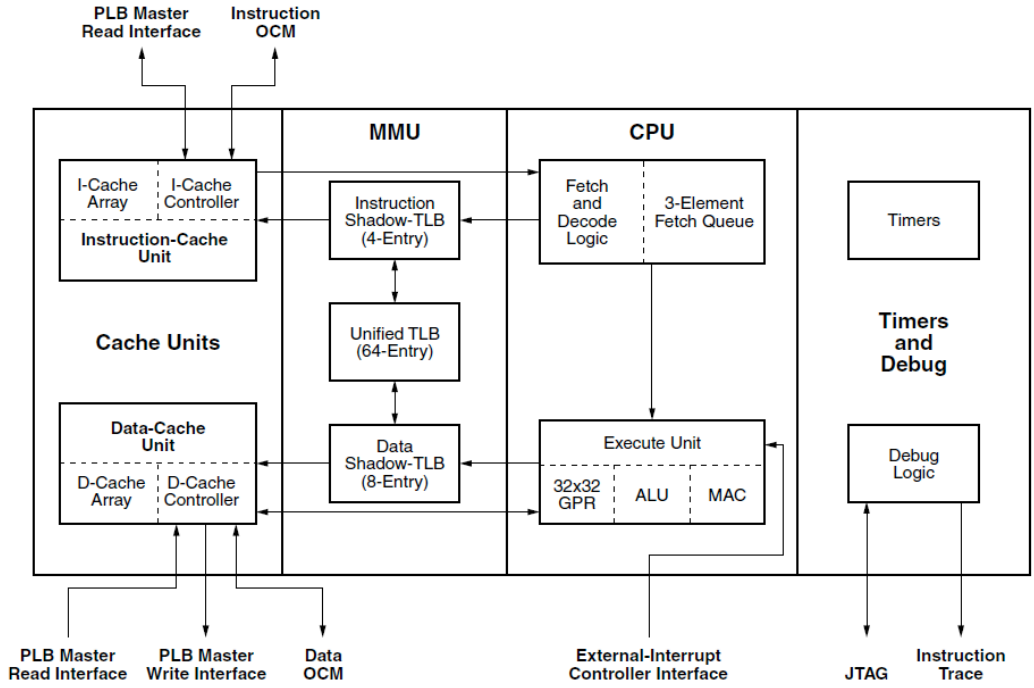
$t_{acc} = t_{MMU} + t_{dato} = (t_{TLB} + f_{TLB}\cdot t_{tablas}) + (t_{L1} + f_{L1}\cdot(t_{L2} + f_{L2}\cdot 4\cdot t_{mem})) = (1 + 0{,}02\cdot 15) + (1 + 0{,}1\cdot(3 + 0{,}2\cdot 4\cdot 10)) = 1{,}3 + 2{,}1 = 3{,}4$ ciclos.

e) Se mejora la integración de caché y TLB, de tal forma que se comience el acceso a caché buscando la etiqueta sin tener la traducción todavía realizada, siempre que esto sea posible. ¿Qué caché(s) se puede(n) integrar de esta forma y por qué?

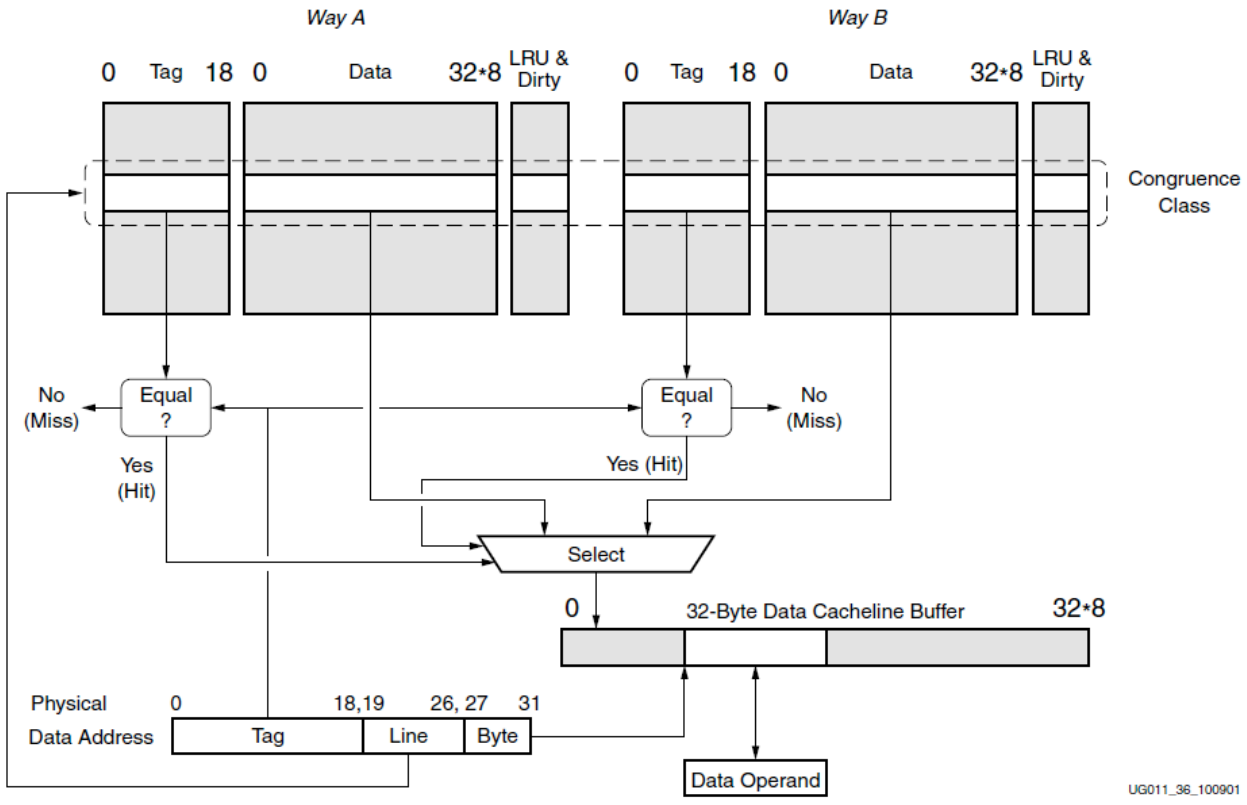
La caché L1 sí se puede integrar de esta forma, ya que índice y B/B son parte del offset, que no hace falta traducir. Por tanto, mientras se busca el marco con el número de página de la dirección virtual, el offset de la dirección virtual (que coincide con el de la dirección real) ya se puede ir utilizando en L1 para buscar la etiqueta. Luego sólo quedará comparar (en caso de acierto en ambas, TLB y caché L1) si el marco coincide con la etiqueta de la caché.

La caché L2 no se puede integrar de esta forma, ya que parte de su índice pertenece al offset y parte al marco, que sólo se conocerá tras realizar la traducción de dirección virtual a real.

3.2.19. PowerPC 405 es un microprocesador muy usado en sistemas empotrados. A continuación se muestra su arquitectura:



PLB y *OCM* son los buses con los que el microprocesador se conecta con las memorias y los periféricos, y *JTAG* e *Instruction Trace* son dos interfaces que se usan para depuración de código. Ambas cachés son similares, teniendo esta arquitectura la de datos:



Tenga en cuenta que es un procesador big endian, por lo que en la figura el bit 0 es el MSB y el bit 31 es el LSB. Finalmente, la MMU se basa en un TLB gestionado por software. El sistema operativo añade hasta 64 entradas al TLB usando unas instrucciones especiales del procesador (p.ej. *tlbwe*, *TLB Write Entry*). Si hay fallo

en el TLB se produce una excepción, que es gestionada por el sistema operativo añadiendo una entrada al TLB para resolver el fallo de página que acaba de ocurrir.

El TLB se compone de tres componentes, Instruction Shadow-TLB, Data Shadow-TLB y Unified TLB, todos completamente asociativos. Si un acceso tanto de instrucciones como de datos tiene acierto en su correspondiente Shadow-TLB, no hay penalización. Si no, se hace la búsqueda en el TLB unificado, con una penalización de 3 ciclos en el caso de acceso de datos o 4 ciclos si es una lectura de instrucción. Si se produce un fallo en el TLB unificado salta una excepción, cuya rutina de atención tarda en media 3500 ciclos en ejecutarse.

a) Responda RAZONADAMENTE a las siguientes preguntas, usando exclusivamente el espacio reservado debajo de cada una

¿La arquitectura del procesador es Harvard o Von Neumann?

En la figura superior de la página anterior se observa con claridad que hay dos caminos diferentes para instrucciones y datos, con buses separados. Es por tanto una arquitectura Harvard.

¿La caché es real o virtual?

La caché es real (de direcciones reales) porque trabaja con direcciones reales, como se puede ver en la figura superior se encuentra después de la MMU, y además en la figura inferior se indica claramente “Physical Data Address”

¿Cuál es el tamaño máximo de la memoria física de datos?

Como se puede ver en la figura inferior, las direcciones físicas de datos son de 32 bits, por lo que puede haber hasta 4 GB de memoria física de datos

¿La caché de datos es de escritura directa o de post-escritura?

La caché es de post-escritura, porque en la figura inferior puede verse que tiene bits de “suciedad” o “dirty” para indicar los bloques que han sido modificados y que todavía no se han escrito en la memoria principal

¿Cuál es el tamaño de la caché y su organización? Indique también el número de bytes por bloque (línea)

Como puede verse en la figura inferior, la caché es asociativa de 2 vías. 8 bits de la dirección se usan como índice para seleccionar la línea, que tiene en total 32 bytes. Por lo tanto, cada vía tiene $2^8 \cdot 32$ Bytes, esto es 8 KB y entonces la caché es de 16 KB.

b) Las siguientes tablas muestran los contenidos del TLB y de las caches de instrucciones y datos. El tamaño de las páginas es de 4 KBytes, y en los bloques de datos de las cachés se sigue una representación big endian (el byte más significativo se corresponde con la dirección más baja). Tanto para el TLB como para las cachés, considerar como vacíos los contenidos no mostrados (indicados con puntos suspensivos). Para facilitar la lectura de los TAGs de las cachés se utiliza una notación "mixta" con dígitos en hexadecimal (negrita) y en binario (tipo de letra más pequeño).

	Instruction Shadow-TLB	
	Pag. Virt.	Marco Real
0	**3FD6B78**	**0ED44**
1	**003556A**	**5344A**
2	**12AC771**	**B126C**
3	**12AC772**	**B126D**

	Unified TLB	
	Pag. Virt.	Marco Real
0	**0A7413E**	**33AF6**
1	**0A7413F**	**33AF7**
2	**0A74140**	**33AF8**
3	**0A74141**	**33AFB**
4	**66FB7A3**	**C6652**
5	**8667224**	**78AAB**
	. . .	. . .
61	**988FDDE**	**98412**
62	**0012AB0**	**B2398**
63	**0012AB1**	**B2399**

	Data Shadow-TLB	
	Pag. Virt.	Marco Real
0	**5477ABE**	**457FC**
1	**0A7413D**	**33AF5**
2	**6455ABC**	**C172B**
3	**7622091**	**A190A**
4	**AB43910**	**7CC6D**
5	**9983F3A**	**43BBD**
6	**789100B**	**ABED3**
7	**AB43911**	**610DC**

Cache Instrucciones				
	VIA 1		VIA 2	
IND	TAG	DATA	TAG	DATA
0	001**9987**	**702C11BE20215A8E04C78D3C158052123EECED491D1D079D2F37E6C73DB6CFD8**	. . .	. . .
. . .	. . .	. . .	. . .	. . .
D5	001**9D7D**	**7942A98A137565B536290564A812998E04797393F4B1C4DCDC75F3043CDD90EA**	. . .	. . .
D6	011**C555**	**6D317495543B6887492215081FC5A082B7F3E74EB76891A8780C66755A7B0951**	. . .	. . .
. . .	. . .	. . .	. . .	. . .
FF	101**8936**	**18207A9B804EF21295549F7674389E90C50CA1F93CDB536792BDDEEA22E631C5**	. . .	. . .

Cache Datos				
	VIA 1		VIA 2	
IND	TAG	DATA	TAG	DATA
0	111**BC34**	**9E3DDA2AAF7BC598588AAE515D40A11E2D5DEB0F04009A3911CFC0133B5BB596**	. . .	. . .
. . .	. . .	. . .	. . .	. . .
47	010**5DF0**	**3CF73A8F107299446B0F3E83DBA49D1309BDC996A07B093802A9FA9518278FAC**	. . .	. . .
48	001**5BBA**	**014A7C42B49D7ABD955F34CEAF723C0E57346B3FA89D3F3AB42E842CCCB82165**	. . .	. . .
. . .	. . .	. . .	. . .	. . .
FF	111**7B31**	**C620871B79DEA8644CCE3C3653257D90A653909AE0694F1CCB5B5351C1495F55**	. . .	. . .

Independientemente de que la arquitectura del procesador sea Harvard o Von Neumann, se supone que hay una memoria principal compartida para instrucciones y datos, con ancho de palabra de 32 bits, de la que se conocen sólo los contenidos de las siguientes direcciones, pero que ocupa todo el espacio de direccionamiento real disponible en el microprocesador:

Dirección	Dato	Dirección	Dato	Dirección	Dato
0B542D94	**A9BC8160**	**55720EC0**	**E87653E4**	**B126DFE8**	**7C3D19CA**
11586B58	**4F611703**	**610DC31C**	**46F6C776**	**C64636B4**	**18990CDA**
47B45DF0	**E8E1D065**	**8DC2E0DC**	**3569CC85**	**D8C8E220**	**E6FE1233**

Para las siguientes operaciones de lectura de instrucciones y datos, obtenga el resultado y el número de ciclos de penalización, teniendo en cuenta que las cachés de datos e instrucciones no penalizan en acierto, pero penalizan 40 ciclos en fallo. Si con los datos proporcionados no se puede saber alguno de los campos (resultado o ciclos), rellene la casilla con una X. SUPONER QUE SON ACCESOS INDEPENDIENTES: Los posibles fallos de TLB y/o caché en un acceso no deben ser tenidos en cuenta para el siguiente (esto es, considerar los contenidos de TLB, cachés y memoria de la página anterior como constantes).

Tipo	Dirección virtual	Resultado	Ciclos
INSTRUCCIÓN	0A74141ABC	7942A98A	4
DATO	8667224AD8	X	43
INSTRUCCIÓN	12AC772FE4	92BDDEEA	0
INSTRUCCIÓN	7622091248	X	X
DATO	AB4391131C	46F6C776	40

c) Un programa típico de 100000 instrucciones se ejecuta en este microprocesador con un CPI=1,4 (sin contar penalizaciones por fallos en TLB y/o cachés) y la frecuencia de operación es de 250 MHz. Calcular el tiempo de ejecución suponiendo que la tasa de fallos local del Shadow-TLB de instrucciones es de 0,032, la del Shadow-TLB de datos es 0,027, la del TLB unificado es 0,015, la de la caché de datos es 0,067 y la de la caché de instrucciones es 0,071. Suponer además que el 15% de las instrucciones son load o stores.

$CPI_{total} = CPI_{ideal} + CDM$

$CDM = F_{Shadow\text{-}TLB\text{-}Instr}(t_{Shadow\text{-}TLB\text{-}Instr} + F_{Unified\text{-}TLB} \cdot t_{Unified\text{-}TLB}) + F_{Cache\text{-}Instr} \cdot t_{Cache\text{-}Instr} +$

$0{,}15 \cdot [\, F_{Shadow\text{-}TLB\text{-}Data}(t_{Shadow\text{-}TLB\text{-}Data} + F_{Unified\text{-}TLB} \cdot t_{Unified\text{-}TLB}) + F_{Cache\text{-}Data} \cdot t_{Cache\text{-}Data} \,] =$

$= 1{,}808 + 2{,}84 + 0{,}15 \cdot (1{,}4985 + 2{,}68) = 5{,}274$

$T_{CPU} = CPI_{total} \cdot NI \cdot T_{Clk} = (1{,}4 + 5{,}274) \cdot 100000 \cdot 4ns = 2{,}67 \text{ ms}$

d) Se mejora el sistema de memoria principal, de tal manera que los fallos en caché de datos y/o instrucciones sólo penalizarán 20 ciclos. Calcular, usando necesariamente la ley de Amdahl, la mejora en el tiempo de ejecución del caso descrito en el apartado anterior.

$F_m = [\, F_{Cache\text{-}Instr} \cdot t_{Cache\text{-}Instr} + 0{,}15 \cdot F_{Cache\text{-}Data} \cdot t_{Cache\text{-}Data} \,] / CDM = 0{,}486$

$A_m = 2$

$A_G = 1 / (1 - F_m + F_m / A_m) = 1{,}32$

El tiempo de ejecución mejorará un 32%, esto es, será ahora 2,67ms / 1,32 = 2,02 ms

3.2.20. El procesador ARM (Advanced RISC Machine) es un RISC de 32 bits ampliamente utilizado en el entorno de los procesadores de aplicaciones específicas (embedded processors). Existen múltiples versiones desde su primera aparición en 1985. En este problema se pretende analizar algunas características de la ejecución del ARM9 representado en la figura.
El procesador ARM9 tiene una segmentación en 5 etapas (fetch, decode, execute, data y write-back); lo que abreviaremos como (I, D, E, M, W) que en ausencia de riesgos y detenciones en memoria finaliza una instrucción por ciclo de reloj. Además tiene implementado adelantamiento de datos de la mejor forma posible.
Una característica interesante de este procesador es el uso de instrucciones que poseen 4 bits en su código de operación que permite hacer su ejecución de forma condicional: si está activa una bandera asociada la instrucción se ejecuta de forma normal, pero si no, se ignora dicha instrucción. Esto permite hacer código más compacto y evitar detenciones por saltos. Como ejemplo el siguiente código C y su conversión a código ensamblador que calcula el máximo común divisor (mcd), según el algoritmo de Euclides.

```
while (i != j){
       if (i > j)
           i -= j;
       else
           j -= i;
    }
```

```
Loop:  CMP    Ri, Rj
       SUBGT  Ri, Ri, Rj
       SUBLT  Rj, Rj, Ri
       BNE    loop
```

```
; set condition "NE" if (i != j),
;               "GT" if (i > j),
;            or "LT" if (i < j)
; if "GT" (greater than), i = i-j
; if "LT" (less than), j = j-i;
; if "NE" (not equal), then loop
```

Es decir, la instrucción de comparación modifica banderas internas (*flags*) del procesador en su ejecución y luego las 2 instrucciones de resta y la de salto sólo se ejecutan si se cumplen las condiciones “mayor que”, “menor que” ó “distinto de” respectivamente.
El código alternativo sin el uso de operaciones condicionales y asumiendo que existen instrucciones de salto condicional por “mayor que” (BGT), por “menor que” (BLT), por “igual” (BEQ) y por “distinto” (BNE) sería:

```
while (i != j){
       if (i > j)
           i -= j;
       else
           j -= i;
    }
```

```
Loop:  BLT  Ri, Rj, menor   ; salta si (i < j)
       SUB  Ri, Ri, Rj      ; resta i = i-j (i > j)
       BEQ  Ri, Ri, Fin     ; salto incondicional
menor: SUB  Rj, Rj, Ri      ; resta j = j-i (i < j)
Fin:   BNE  Ri, Rj, loop    ; salta si no iguales
```

a) Suponiendo que todos los saltos condicionales resuelven la condición y calculan la dirección de salto en la etapa *execute* (E) y que se utiliza predicción no efectiva, ¿cuántos ciclos se pierden en caso de salto efectivo y no efectivo? Justifique su respuesta usando las tablas adjuntas. El nemónico BXX indica cualquier tipo de salto condicional.

Si Salto Efectivo: Pierde **2** ciclos	Si Salto NO Efectivo: Pierde **0** ciclos

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
BXX	I	D	E[1]	M	W					
Next		I	D	-						
N+1			I	-						
N+2										
T				I	D	E	M	W		
T+1					I	D	E	M	W	

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
BXX	I	D	E[1]	M	W					
Next		I	D	E	M	W				
N+1			I	D	E	M	W			
N+2				I	D	E	M	W		
T										
T+1										

b) Cuántos ciclos de reloj necesitará la ejecución del código que utiliza restas condicionales (SUBGT y SUBLT) y el que no las utiliza para el caso i=35; j=24 que hace ejecutar el ciclo principal 6 veces con i>j y 3 veces con j>i. Para justificar sus respuestas utilice las tablas adjuntas

b.1: Cantidad de ciclos del código 1:
9 iteraciones de 6 ciclos, más dos ciclos última iteración (ó -2 si no se considera vaciar el pipeline)
9 x 6 + 2 = 56 ciclos ó 9 x 6 - 2 = 52 (es aceptado como correcto, incluso sin sumar ó restar los 2 ciclos)
En realidad este algoritmo necesita una iteración extra para salir donde i=j tras la última actualización.
El resultado correcto es 10 x 6 + 2 = 62 ciclos ó 10 x 6 - 2 = 58 si no se considera el vaciado del pipeline

b.2: Cantidad de ciclos del código 2:
6 iteraciones de 8 ciclos, 3 iteraciones de 7 ciclos más dos ciclos última iteración para el vaciado ó -2 si no se considera el vaciado
6 x 8 + 3 x 7 + 2 = 71 ciclos ó 6 x 8 + 3 x 7 - 2 = 67 ciclos sin el vaciado.

Justificación de ciclos utilizados para la ejecución del código con instrucciones condicionales

CMP	I	D	E	M	W							
SUBGT		I	D	E	M	W						
SUBLT			I	D	E	M	W					
BNE				I	D	E	M	W				
N+1					I	D	-					
N+2						I	-					
CMP							I	D	E			
SUBGT								I	D			
…	1	2	3	4	5	6						

Siempre compara y pone los "flags". Las instr. siempre se ejecutan y efectúa el computo según el valor de los flags.

dos ciclos perdidos si salto ef
Comienza nuevo ciclo

Se usan 6 ciclos por iteración

Justificación de ciclos utilizados para la ejecución del código sin instrucciones condicionales (i<j)

BLT (ef)	I	D	E	M	W								
SUB		I	D	-									
BEQ			I	-									
SUB				I	D	E	M	W					
BNE (ef)					I	D	E	M	W				
N+1						I	D	-					
N+2							I	-					
BGT								I	D	E	M	W	
SUB									I	D	E/-	M/-	W/-
…	1	2	3	4	5	6	7						

1er salto efectivo

Este salto no se ejecuta

Si salto efectivo se pierden dos ciclos adicionales

Comienza nuevo ciclo

Se usan 7 ciclos por iteración E/- indica que depende de si BGT es efectivo ó no si se pasa por las etapas E, M y W de la siguiente instrucción.

Justificación de ciclos utilizados para la ejecución del código sin instrucciones condicionales (i>j)

BLT (ne)	I	D	E	M	W								
SUB		I	D	E	M	W							
BEQ (ef)			I	D	E	M	W						
SUB				I	D	-							
BNE					I	-							
BNE (ef)						I	D	E	M				
N+1							I	D	-				
N+2								I	-				
BGT									I	D	E	M	W
SUB										I	D	E/-	M/-
…	1	2	3	4	5	6	7	8					

1er salto no efectivo

Este salto se ejecuta

este salto se carga 2 veces si salto efectivo se pierden dos ciclos adicionales

Comienza nuevo ciclo

Se usan 8 ciclos por iteración

c) Se agregan a este procesador memorias caché. La caché de instrucciones es de 32 kBytes organizada en 64 bytes por bloque y asociativa de 8 vías. Recordar que es un procesador de 32 bits donde las palabras, direcciones e instrucciones son de ese tamaño.

C1. ¿Cómo se divide la dirección para el acceso a memoria caché?
Total Bloques: Tamaño Total / Tamaño bloque = 32 KB / 64 B = 512 bloques.
Como es de 8 vías hay un total de 512 bloques / 8 vias = 64 entradas (6 bits para direccionarla)

Cada bloque posee 64 bytes (16 palabras de 4 bytes) luego el campo “Byte en Bloque” tiene 6 bits

El resto de bits 32 – 6 – 6 = 20 bits de etiqueta.

Etiqueta	Índice	Byte en Bloque
20	6	6

C2. Suponiendo que el código ensamblador que utiliza restas condicionales (SUBGT y SUBLT) está escrito a partir de la posición de memoria 0000_8034 (hexadecimal), que el programa no está presente en la caché al iniciar el análisis y que durante su ejecución no se ejecuta otro programa concurrentemente, ¿cuántos fallos de caché se producen? Justifique su respuesta.

El programa ocupa 4 instrucciones que comienzan en: 0000_8034, 0000_8038, 0000_803C y 0000_8040, como la última es un salto también se capturan las dos siguientes (0000_8044 y 0000_8048), antes de actualizar el contador de programas.

La división en los campos de acceso nos da:

Dir	tag	índice	BB	1ra vez	Resto Iterac
0000_8034	00008	000	114	F	A
0000_8038	00008	000	118	A	A
0000_803C	00008	000	11C	A	A
0000_8040	00008	001	000	F	A
0000_8044	00008	001	004	A	A
0000_8048	00008	001	008	A	A

Es decir ocuparán dos entradas diferentes. En una de las 8 vías de la entrada 0 (con 6 bits) estarán las 3 primeras instrucciones, en tanto que en una de las 8 vías de la entrada 1 (con 6 bits) estarán la última instrucciones del programa y las otros dos que se capturan innecesariamente.
Esto provocará un total de **dos fallos** en la primera ejecución y ningún otro a menos que otros procesos ó programas desaloje las 8 entradas de cada una de las vías.