Es demana:

a) Quin serà el temps mig d'accés **Tma** per cada configuració (en cicles)?
b) Quin serà el temps d'execució **Texec** de 1 instrucció real en cada cas?
c) Per quina opció optaríeu i perquè? (en una línia)
d) Creus que es pot trobar alguna opció millor en base a les **dades de que disposem**? En cas afirmatiu, digues quina i perquè.

Problema 7. Repaso Memoria Virtual

Dado el siguiente código escrito en ensamblador x86:

```
      movl $0, %ebx
      movl $0, %esi
for:  cmpl $512*1000, %esi
      jge end
(a)   movl (%ebx, %esi, 4), %eax
(b)   addl %eax, 8*1024(%ebx, %esi, 4)
(c)   movl %eax, 16*1024(%ebx, %esi, 4)
      addl $512, %esi
      jmp for
end:
```

Suponiendo que la memoria utiliza **páginas de tamaño 8KB** y que utilizamos un **TLB de 4 entradas (reemplazo LRU)**, responde a las siguientes preguntas:

a) Para cada uno de los accesos (etiquetas a, b, c), indica a qué página de la memoria virtual se accede en cada una de las 17 primeras iteraciones.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4
b	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4	5
c	2	2	2	2	3	3	3	3	4	4	4	4	5	5	5	5	6

b) **Calcula** la cantidad de **aciertos de TLB**, en todo el bucle: . . . . . 3748 . . . . . .
c) **Calcula** la cantidad de **fallos de TLB**, en todo el bucle: . . . . . 252 . . . . . .

Suponiendo que la memoria utiliza **páginas de tamaño 4KB** y que utilizamos un **TLB de 4 entradas (reemplazo LRU)**, responde a las siguientes preguntas:

d) Para cada uno de los accesos (etiquetas a, b, c), indica a qué página de la memoria virtual se accede en cada una de las 17 primeras iteraciones.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8
b	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10
c	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11	12

e) **Calcula** la cantidad de **aciertos de TLB**, en todo el bucle: . . . . . 2500 . . . . . .
f) **Calcula** la cantidad de **fallos de TLB**, en todo el bucle: . . . . . 1500 . . . . . .

Problema 8. Repaso Memoria Virtual

Tenim un processador amb memòria virtual basada en paginació. El sistema de memòria virtual te les següents característiques:

- 16 bits d'adreça lògica
- 15 bits d'adreça física
- mida de pàgina 8 KB
- reemplaçament LRU

El contingut de la taula de pàgines es mostra a la figura 1, on VPN = número de pàgina lògica, P=bit de presencia, M=pàgina modificada i PPN=número de pàgina física. El contingut de la memòria física es mostra a la figura 2. En cas que la memòria física s'empleni i faci falta reemplaçar una pàgina es segueix un algorisme LRU. De les tres pàgines inicialment presents a memòria, la pàgina física 1 (lògica 3) es la que fa menys temps que ha estat accedida, la següent es la 0 (lògica 2) y, finalment, la 2 (lògica 4) es la que fa més temps que ha estat accedida.

1 Contingut **inicial** de la **Taula de Pàgines**

VPN	P	M	PPN
0	0	0	-
1	0	0	-
2	1	0	0
3	1	0	1
4	1	0	2
5	0	0	-
6	0	0	-
7	0	0	-

2 Contingut **inicial** de **Memòria**

pàgina física	pàgina lògica
0	2
1	3
2	4
3	-

3 Contingut **final** de la **Taula de Pàgines**

VPN	P	M	PPN
0	1	0	0
1	0		
2	1	1	1
3	1	0	3
4	1	1	2
5	0		
6	0		
7	0		

4 Contingut **final** de **Memòria**

pàgina física	pàgina lògica
0	0
1	2
2	4
3	3

a) **Empleneu** la següent taula indicant, per cada referència, la pàgina lògica (VPN), el desplaçament, l'adreça física resultant de la traducció. Indiqueu amb una creu (X) quan es produeix un fallo de pàgina, quan es llegeix de disc dur, quan s'escriu a disc dur i, en cas de reemplaçar una pàgina, indiqueu el VPN i PPN. Indiqueu també el contingut final de la taula de pàgines i de la memòria física (figures 3 i 4)

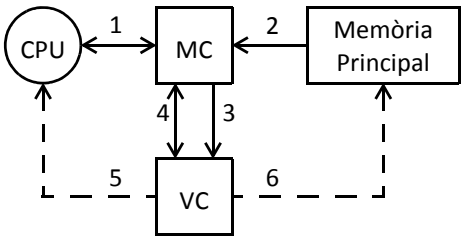
adreça lògica (hexa)		**VPN (hexa)**	**desplaçament (hexa)**	**adreça física (hexa)**	**fallo de pàgina**	**lectura disc**	**escriptura disc**	**Pàgina reemplaçada**	
								VPN	**PPN**
escriptura	F458	7	1458	7458	X	X			
escriptura	8666	4	0666	4666					
lectura	1BBF	0	1BBF	1BBF	X	X		2	0
escriptura	5C44	2	1C44	3C44	X	X		3	1
lectura	6600	3	0600	6600	X	X	X	7	3
lectura	4000	2	0000	2000					

Problema 9. Caches petites y simples

Hem vist a teoria que una cache d'emplaçament directe te un temps d'accés inferior a una associativa. Una cache directa també sol tenir una taxa de fallades més elevada degut als conflictes. Una possible sol.lució al problema es tenir el que s'anomena una cache de víctimes (*Victim Cache*) que permet reduir les fallades degudes a conflictes. La idea es una aplicació del concepte de cache petita i simple.

Una cache de víctimes (VC) es una cache totalment associativa, però molt petita (4-8 blocs) que treballa en paral.lel a la memòria cache (MC), que habitualment es d'emplaçament directe. La VC emmagatzema aquells blocs que han estat expulsats de la MC. Encara que la VC es totalment associativa el seu temps d'accés es similar o inferior al de la MC (directa) perquè es molt petita.

La figura il·lustra el funcionament de un sistema de memòria amb cache de víctimes. Quan hi ha un accés a memòria es poden donar les següents situacions:



- *hit* a MC: es serveix la dada de MC (1) (no penalització).
- *miss* a MC (i també a VC): es porta el bloc corresponent de memòria principal (2) a MC, i el bloc expulsat de la MC s'emmagatzema a VC (3) (penalització molt elevada).
- *miss* a MC però *hit* a VC: donat que el bloc demanat es troba a VC no es necessari accedir a memòria principal sinó que l'obtenim directament de VC i el bloc expulsat de MC

l'emmagatzemem a VC, es a dir intercanviem el bloc demanat i el expulsat entre MC i VC (4) (penalització molt baixa).

Donat que la VC i la MC es poden consultar en paral.lel, una possible implementació podria permetre que en el darrer cas (*miss* a MC i *hit* a VC) la dada es servis a la CPU directament per la VC (5) amb el que es podria aconseguir que no hi hagués cap penalització, tot i que la circuiteria necessària per poder-ho fer (multiplexors, comprovació en paral.lel de les dos caches, ...) podria incrementar lleugerament el temps d'accés. Una altra consideració a tenir en compte es el dels blocs modificats en una cache *copy back*. Donat que quan un bloc es expulsat de MC no desapareix del sistema, sinó que es copia a VC, només serà necessari actualitzar la memòria principal quan un bloc modificat sigui expulsat definitivament de VC (6).

Aquest problema ens permetrà veure el funcionament de les caches de víctimes i els seus avantatges. Per construir una memòria cache s'han considerat tres possibilitats:

- Una memòria cache d'emplaçament directe amb 8 blocs.
- Una memòria cache associativa per conjunts amb 4 conjunts de 2 blocs cadascun i amb reemplaçament LRU.
- Una memòria cache d'emplaçament directe amb 8 blocs a la que s'ha afegit una *victim cache* amb reemplaçament FIFO de 2 blocs de capacitat.

a) **Indiqueu** quins accessos seran *hit* (amb una X) per cada una de les tres possibilitats per la següent seqüència de **referències a bloc** (en octal) on tots els accessos son lectures. En el cas de la directa + VC es considerarà *miss* si el bloc referenciat no es troba ni a MC ni a VC.

Bloc de memôria	**73**	**55**	**43**	**45**	**73**	**45**	**13**	**43**	**73**	**55**	**45**	**73**	**15**	**43**
Directa						X						X		
2-associativa					X	X				X	X	X		X
Directa + VC					X	X		X	X		X	X		X

b) Creus que hi hauria cap diferencia si la VC fes servir un reemplaçament LRU? Perquè? No, perquè la línia que porta més temps dins la VC també és la que porta més temps sense fer-se servir, ja que quan una línia és a la VC però no a la MC, s'intercanvien.

Volem estudiar la implementació d'aquestes 3 caches com a cache de dades en un processador. Per un programa P que només fa lectures executat en aquest processador amb memòria ideal (on tots els accessos a memòria tarden 1 cicle) sabem que ha executat $10x10^9$ instruccions en $12x10^9$ cicles i s'han fet $3x10^9$ accessos a memòria (dades).

c) **Calculeu** el CPI amb memòria ideal (CPI_{ideal}). $CPI = \frac{12 \cdot 10^9}{10 \cdot 10^9} = \frac{12}{10} = 1,2$

d) **Calculeu** el ratio *nr* (accessos a memòria per instrucció). $nr = \frac{3 \cdot 10^9}{10 \cdot 10^9} = \frac{3}{10} = 0,3$

La cache es troba al camí crític del processador i volem que en cas d'encert es pugui llegir la dada en 1 sol cicle, de forma que el temps d'accés a la cache ens determinarà el temps de cicle del processador en totes les implementacions.

Cache d'**emplaçament directe**: El temps de cicle (Tc) es de 10 ns/cicle, la taxa de fallades (m) es de 0.1 fallades/accés i el temps de penalització en cas de fallada (Tpf) es de 10 cicles.

e) Quants cicles tarda en executar-se el programa P? $1,5 \cdot 10^{10}$ cicles

f) Quin es el temps d'execució de P? (en segons) $1,5 \cdot 10^{10} \cdot 10\ ns = 1,5 \cdot 10^{9}\ s = 150\ s$

Cache **2-associativa**: El temps de cicle (Tc) es de 12 ns/cicle, la taxa de fallades (m) es de 0.05 fallades/accés i el temps de penalització en cas de fallada (Tpf) es de 9 cicles.

g) Perquè creus que el temps de penalització en cas de fallada es de 9 cicles mentre que en el cas d'emplaçament directe era de 10 cicles si sabem que la memòria principal es la mateixa?

h) Quants cicles tarda en executar-se el programa P? $10 \cdot 10^9 \cdot (1,2 + 0,3 \cdot 0,05 \cdot 0,9) = 1,135 \cdot 10^{10}$ cicles

i) Quin es el temps d'execució de P? (en segons) $1,135 \cdot 10^{10} \cdot 12 \cdot 10^{-9} = 160,2\ s$

Cache **emplaçament directe + victim cache** amb accés simultani: A pesar que la victim cache es més ràpida que la d'emplaçament directe, la lògica i el multiplexor necessaris per controlar de quina cache obtindrem la dada fa que el temps d'accés sigui lleugerament més alt que en el cas de la cache directa. El temps de cicle (Tc) es de 11 ns/cicle, la taxa de fallades (m) global del conjunt MC+VC es de 0.06 fallades/accés i el temps de penalització en cas de fallada (Tpf) es de 10 cicles.

j) Quants cicles tarda en executar-se el programa P? $10 \cdot 10^9 \cdot (12 + 0,3 \cdot 0,06 \cdot 10) = 1,38 \cdot 10^{10}$ cicles

k) Quin es el temps d'execució de P? (en segons) $1,38 \cdot 10^{10} \cdot 11 \cdot 10^{-9} = 151,8\ s$

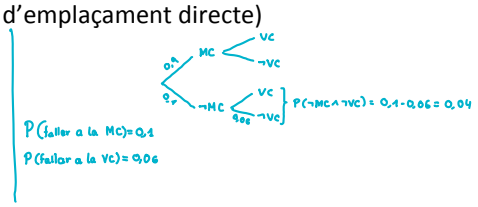
Cache **emplaçament directe + victim cache** amb accés seqüencial: En aquesta segona implementació, els accessos que s'han de fer a la victim cache tenen una penalització addicional de un cicle, però el temps de cicle es el de la cache d'emplaçament directe. El temps de cicle (Tc) es de 10 ns/cicle, la taxa de fallades (m) global del conjunt MC+VC es de 0.06 fallades/accés, el temps de penalització en cas que fallem a MC però encertem a VC (Tpvc) es de 1 cicle i el temps de penalització en cas que fallem a totes dues (Tpf) es de 11 cicles.

l) Perquè creus que el temps de penalització en cas de fallada es de 11 cicles mentre que en el cas d'emplaçament directe era de 10 cicles si sabem que la memòria principal es la mateixa? Perquè els accesos a la VC tenen un cicle extra.

m) Calcular la probabilitat que un accés falli a MC però encerti a VC? (pista: es pot deduir a partir de la taxa de fallades global i la taxa de fallades que tenim quant només hi ha la cache d'emplaçament directe)

n) Quants cicles tarda en executar-se el programa P? $1{,}41 \cdot 10^{10}$ cicles

o) Quin es el temps d'execució de P? (en segons) $1{,}41 \cdot 10^{10} \cdot 10 \cdot 10^{-9} = 141$ s



Problema 10. Predicción de vía

Una CPU funciona a un voltaje de 1,2 V y una frecuencia de 2GHz. Se ha determinado que esta CPU tiene una corriente de fugas de 3 A y que a pleno rendimiento tiene una carga capacitiva equivalente de 5 nF (nanoFaradios).

a) **Calculad** la potencia media dinámica (debida a conmutación), la potencia media estática (debida a fugas) y la potencia media total.

Se desea integrar esta CPU con una memoria cache de datos 2-asociativa de 128 KB de capacidad y un tamaño de bloque de cache de 64 bytes. Las direcciones generadas por la CPU son de 48 bits. La corriente de fugas de la memoria RAM estática es de 3 μA (microAmperios) por bit. La energía consumida durante un acceso a la memoria de etiquetas es de 5 nJ (nanoJoules) por vía y la consumida durante un acceso a la memoria de datos es de 25 nJ por vía.

b) **Calculad** el numero de conjuntos, el de bloques de cache, el de vías y el de bloques por vía.

c) **Dibujad** una dirección indicando claramente los campos usados para seleccionar el byte dentro del bloque, seleccionar el conjunto de la cache y los bits usados como etiqueta.

d) **Calculad** el tamaño **en bits** de la memoria de datos y el de la memoria de etiquetas de una vía (por simplicidad ignoraremos el bit de validez y otros bits de control).

e) **Calculad** la potencia media estática (debida a fugas) de la cache.

Se desean comparar diversas implementaciones alternativas de cache de datos 2-asociativa: **paralela**, **serie**, y con **predictor de vía**. Para compararlas se usa un *benchmark* con 4×10^9 instrucciones dinámicas que realiza 10^9 accesos de datos a memoria y 2×10^9 operaciones aritméticas de punto flotante. Este *benchmark* tiene un 10% de fallos en la cache descrita anteriormente.

En la implementación **paralela**, se accede simultáneamente tanto a las memorias de etiquetas como las de datos de ambas vías. Un acceso a cache se realiza en 1 ciclo y la penalización media por fallo de cache es de 20 ciclos. El *benchmark* ejecutado con la implementación **paralela** de la cache ha tardado 5 segundos.

f) **Calculad** los MFLOPS de la implementación **paralela.**

g) **Calculad** el CPI de la implementación **paralela** y el CPI que obtendríamos con una memoria ideal (CPI_{ideal}) en donde todos los accesos tardan 1 ciclo.

h) **Calculad** la energía dinámica consumida por un acceso a la cache. Para simplificar asumiremos que todos los accesos consumen lo mismo sean acierto o fallo, la energía extra consumida en acceder a memoria principal en caso de fallo está fuera de los objetivos de este problema.

i) **Calculad** la potencia (dinámica) media consumida en acceder a la cache

j) **Calculad** la potencia media total (estática+dinámica) consumida por el sistema CPU-cache.

k) **Calculad** la energía total consumida para ejecutar el *benchmark* y la eficiencia en MFLOPS/Watt.

En la implementación **serie** un acceso tarda 2 ciclos. En el primer ciclo se accede a las memorias de etiquetas de ambas vías. Una vez determinada la vía que contiene el dato, en el segundo ciclo se accede solo a la memoria de datos de dicha vía. La penalización en caso de fallo sigue siendo de 20 ciclos ya que en el primer ciclo del acceso ya se puede determinar si es acierto o fallo. Obsérvese que respecto la implementación **paralela**, los aciertos tienen una penalización de 1 ciclo.

l) **Calculad** el tiempo de ejecución y los MFLOPS de la implementación **serie.**

m) **Calculad** la energía consumida por un acceso a la cache.