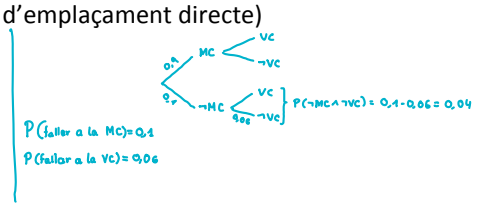
Cache **emplaçament directe + victim cache** amb accés seqüencial: En aquesta segona implementació, els accessos que s'han de fer a la victim cache tenen una penalització addicional de un cicle, però el temps de cicle es el de la cache d'emplaçament directe. El temps de cicle (Tc) es de 10 ns/cicle, 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.

n) **Calculad** la potencia (dinámica) media consumida en acceder a la cache
o) **Calculad** la potencia media total consumida por el sistema CPU-cache.
p) **Calculad** la energía total consumida para ejecutar el *benchmark* y la eficiencia en MFLOPS/Watt.

En la implementación con **predictor de vía**, este predice la vía probable en que se encuentra el dato y se accede solo a las memorias de etiquetas y de datos de esa vía. En caso de fallo del predictor hay que acceder a la otra vía (memorias de etiquetas y datos). El predictor usado consiste en una memoria de 8k x 1 bits indexado con los bits bajos del PC de las instrucciones de acceso a memoria. Este predictor tiene una tasa de aciertos del 80% del total de accesos a memoria y cada acceso al predictor consume 1nJ. En esta implementación se pueden dar las siguientes situaciones:

- El predictor acierta: se accede a 1 sola vía (datos+etiquetas) en 1 ciclo (no hay penalización) y al comprobar los tags se comprueba que es acierto de cache.
- El predictor falla pero es acierto de cache: El acceso tarda 2 ciclos, en el primero se accede a la vía probable (aunque equivocada) al comprobar los tags se descubre que es fallo (de via) y en el segundo se accede a la vía correcta y se descubre que es acierto de cache (1 ciclo de penalización).
- El predictor falla y además es fallo de cache: En el primer ciclo se accede a la vía probable (aunque incorrecta), en el segundo ciclo se accede a la otra vía y se descubre que es fallo de cache (con lo que hay que acceder a memoria principal). Obsérvese que en este caso la penalización es de 21 ciclos ya que no se descubre el fallo de cache hasta que se accede a la 2a vía.

q) ¿Puede darse al caso de que un acierto del predictor de vía sea fallo de cache? ¿porqué?
r) **Calculad** la potencia media estática (debida a fugas) del predictor y compárala con la de la cache (se calcula de la misma forma ya que se ha empleado el mismo tipo de memoria estática).
s) **Calculad** el tiempo de ejecución y los MFLOPS de la implementación con **predictor de vía.**
t) **Calculad** la energía consumida por un acceso en que el predictor acierta y uno en que el predictor falla (tener en cuenta la energía consumida por el acceso al predictor). Calcular también la energia media consumida por acceso.
u) **Calculad** la potencia (dinámica) media consumida en acceder a la cache
v) **Calculad** la potencia media total consumida por el sistema CPU-cache (acuérdate de las fugas del predictor).
w) **Calculad** la energía total consumida para ejecutar el *benchmark* y la eficiencia en MFLOPS/Watt.
x) **Calculad** la ganancia en eficiencia energética de la implementación serie sobre la paralela y la de predicción de vía sobre la serie.

Problema 11. Caches segmentadas

En un procesador X el camino crítico, y por tanto el tiempo de ciclo, está limitado por la memoria cache de datos. El tiempo de los componentes de la memoria cache de datos se desglosa de la siguiente forma:

Componente	Tiempo
Memoria de etiquetas	0,30 ns
Selección de vía	0,15 ns
Memoria de datos	0,45 ns
Mux/Driver de datos	0,10 ns
Registro de desacoplo (en caso que se use)	0,05 ns

Queremos evaluar el rendimiento de 4 posibles implementaciones de la memoria cache para el procesador X:

- Procesador **X1**: La cache de datos tiene una implementación paralela y el tiempo de acceso a la cache es de 1 ciclo de procesador
- Procesador **X2**: La cache de datos está segmentada en 2 etapas y el tiempo de acceso a la cache es de 2 ciclos de procesador
- Procesador **X3**: La cache de datos está segmentada en 3 etapas y el tiempo de acceso a la cache es de 3 ciclos de procesador

- Procesador **X4**: La cache de datos está segmentada en 4 etapas y el tiempo de acceso a la cache es de 4 ciclos de procesador

a) **Calculad** el tiempo de ciclo de la cache de datos y el tiempo total de un acceso para los procesadores X1, X2, X3 y X4, usando la distribución mas adecuada de los componentes por etapas.

b) **Razonad** porque descartamos las opciones X2 y X4 para el resto del problema

c) **Calculad** la frecuencia de reloj de los procesadores X1 y X3

Un programa P que ejecuta $2x10^9$ instrucciones tiene un 60% de instrucciones aritméticas, un 20% de instrucciones de salto y un 20% de instrucciones de acceso a memoria (Load/Store). Las instrucciones aritméticas tardan 5 ciclos, las de salto 4 y las de memoria 4 ciclos + los ciclos del acceso a la cache.

d) **Calculad** el CPI del programa P para los procesadores X1 y X3 suponiendo que nunca hay fallos en la cache de datos.

e) **Calculad** el speedup de X3 sobre X1 en % suponiendo que nunca hay fallos en la cache de datos

Sabemos que el programa P tiene un 10% de fallos en la cache de datos utilizada y que el tiempo de penalización en ambos casos es de 60 ciclos.

f) **Calculad** el speedup real de X3 sobre X1 en % teniendo en cuenta la jerarquía de memoria completa.

Problema 12. Cache no bloqueante.

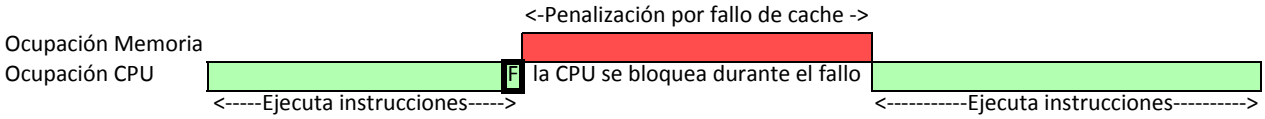
Nota: Conviene desenpolvar los apuntes de Probabilidad y Estadistica y repasar las distribuciónes de probabilidad geometrica y uniforme discreta.

Se ha simulado la ejecución de un programa P en un procesador que denominaremos IDEAL. En este procesador IDEAL no hay ninguna penalización por fallo de cache. De esta simulación se ha obtenido que el programa P se ha ejecutado en $5x10^9$ ciclos durante los que ha ejecutado $2x10^9$ instrucciones, de las que $500x10^6$ son instrucciones de acceso a datos (Load/Store) y se han producido $50x10^6$ fallos en la cache de datos (los fallos en la cache de instrucciones son negligibles, con lo que los ignoraremos durante todo el problema). Suponemos que la probabilidad de fallar en cualquier ciclo es la misma y es independiente de que se haya fallado o no en el ciclo anterior, por lo que el número de ciclos entre fallos sigue una ***distribución geométrica***.

a) **Calculad** el CPI de P en el procesador IDEAL (CPI_{IDEAL})

b) **Calculad** el número medio de ciclos transcurridos entre 2 fallos.

El mismo programa lo ejecutamos en un procesador real con las mismas características que el IDEAL con la única diferencia que en caso de fallo en la cache de datos se bloquea la ejecución de instrucciones durante un cierto numero de ciclos que corresponden al tiempo de penalización por fallo de cache (Tpf) hasta que se resuelve el fallo. La siguiente figura ilustra este hecho cuando se detecta un fallo (F).

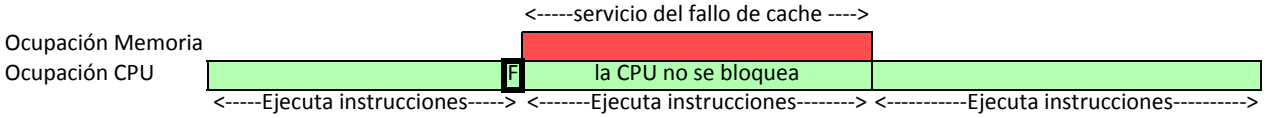


El programa P se ha ejecutado en el procesador con cache bloqueante (que llamaremos procesador B) en 4 segundos. Este procesador B funciona a una frecuencia de 2 GHz.

c) **Calculad** el CPI de P en el procesador B (CPI_B)

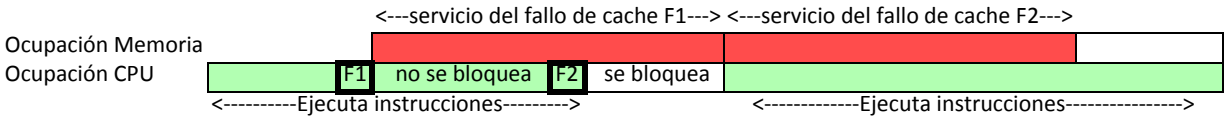
d) **Calculad** el tiempo de penalización por fallo de cache (Tpf) en ciclos.

El rendimiento del procesador se puede mejorar implementando una cache no bloqueante (con las mismas características de tamaño de bloque, asociatividad, reemplazo, etc) de forma que cuando se produce un fallo de cache el procesador siga ejecutando instrucciones tal como muestra la siguiente figura.



En la implementación elegida (que denominamos procesador N) dispondremos de un único MSHR (Miss Holding Status Register) que nos permite tener como máximo un fallo pendiente. Si durante el servicio de este fallo (F1) se

produce un segundo fallo (F2), hay que esperar a que la jerarquía de memoria complete el fallo en servicio antes de que pueda servir el siguiente fallo, con lo que el procesador se bloqueará por unos ciclos, tal como muestra la siguiente figura. Este mecanismo en que se permite un único fallo pendiente se denomina "hit under miss".



Durante la fase en que la CPU ejecuta instrucciones estas se ejecutan con la misma distribución que en el procesador IDEAL, por lo que el número medio de ciclos entre F1 y F2 será el mismo.

e) **Calculad** la probabilidad de que se produzca un segundo fallo durante el servicio de un fallo anterior

f) ¿Puede producirse un tercer fallo?

Si se produce un segundo fallo durante el intervalo de servicio de un fallo anterior, este se puede producir en cualquiera de los ciclos que dura el servicio, con la misma probabilidad. Es decir, se trata de una distribución de probabilidad ***uniforme discreta*** (dado de 60 caras).

g) **Calculad** cuantos ciclos se pierden como máximo y como mínimo en función de en que ciclo del intervalo se produce el segundo fallo.

h) **Calculad** el número medio de ciclos perdidos debido al segundo fallo (repasa cual es el valor medio esperado en una distribución de probabilidad uniforme discreta, o sea un dado numerado de 0 a 59)

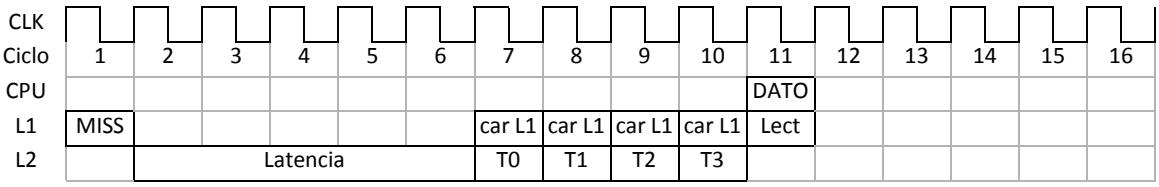
i) **Calculad** el numero de ciclos necesario para ejecutar P en el procesador N (con cache no bloqueante)

Debido a la complejidad añadida de la cache no bloqueante, el procesador N funciona a un frecuencia ligeramente inferior de 1,9 GHz

j) **Calculad** la ganancia (speedup) del procesador N sobre el B

Problema 13. Continuación anticipada, Transferencia en desorden

Un procesador tiene una cache de primer nivel (que llamaremos L1) con bloques de 32 bytes y está conectado a un segundo nivel de jerarquía de memoria (que llamaremos L2) mediante un bus de 8 bytes de ancho. El primer nivel (L1) tiene un tiempo de acceso de 1 ciclo cuando se produce un acierto. Cuando fallamos en el primer nivel accedemos al segundo nivel (L2) para leer un bloque de datos. Suponemos que solo se realizan lecturas y que nunca hay fallos en L2. El siguiente cronograma ilustra un fallo en L1: se necesitan 5 ciclos de latencia y 4 para transferir los datos (T0-T3). Los datos se cargan en L1 mientras se transfieren (car L1), y una vez tenemos todo el bloque en L1, hay que realizar una lectura en L1 (Lect) para enviar el dato a la CPU (DATO), cosa que ocurre dentro del mismo ciclo.



Al ejecutar un programa en un simulador, hemos obtenido que, a una frecuencia de 2GHz el programa tardaría 2 segundos en el caso ideal de que no haya fallos de cache, que se han realizado 10^9 accesos a memoria y que el 20% de los accesos provocarían un fallo en L1.

a) **Calculad** el tiempo de ciclo y los ciclos que tarda el programa en el caso ideal.

b) **Calculad** los ciclos de penalización de un fallo de L1 y el tiempo de ejecución del programa teniendo en cuenta la penalización debida a los fallos de L1.

Para mejorar el rendimiento, permitimos que el procesador pueda captar el dato en el mismo ciclo que se transfiere de L2 a L1 (continuación anticipada o *early restart*). Mediante simulación sabemos que cuando se produce un fallo en L1 en nuestro programa, en el 70% de los casos el dato deseado se corresponde al que se transfiere en el ciclo T0,