Ricard Guixaró Trancho

Problema 6. Repaso Cache

L'empresa A.C.M.E. esta dissenyant un nou processador del que sabem que genera **1.3 referències** a memòria per instrucció de les que **0.3 son de dades**. Els seus dissenyadors s'han adonat de que al mateix xip queda espai suficientper posar-hi un poc de cache. Després de fer alguns càlculs resulta que poden posar-hi una cache de 16k o dues (una de 8k i una de 4k), i han optat per una de les dues configuracions següents:

- Una sola cache **unificada de 16Kb**. En aquest cas (si la cache fos ideal) tenim un **CPI_{ideal} de 1.5** cicles/instrucció degut a que quan accedim a la cache per buscar una dada no podem buscar una instrucció al mateix temps.
- Dues caches separades, una d'**instruccions de 4Kb** i una de **dades de 8Kb.** En aquest cas el **CPI_{ideal} es de 1.2** cicles/ instrucció ja que podem buscar dades i instruccions simultaniament.

En qualsevol de les dues configuracions:

- Tc (temps de cicle): 10ns
- Tsa (temps de servei en cas d'encert): 1cicle.
- Tsf (temps de servei en cas de fallada): 10 cicles.

La taxa de fallades, per diferents configuracions de cache, es mostra a la següent taula:

Mida	Instruccions	Dades	Unificada
4K	8.6%	8.7%	11.2%
8K	5.8%	6.8%	8.3%
16K	3.6%	5.3%	5.9%

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?

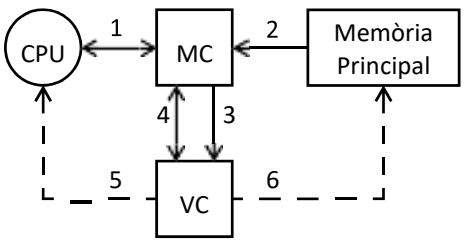
d) Creus que es pot trobar alguna opció millor en base a les **dades de que disposem**? En cas afirmatiu, digues quina i perquè.

Problema 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. Laidea 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 laMC (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 lessegüents 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 moltbaixa).

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														
2-associativa														
Directa + VC														

b) Creus que hi hauria cap diferencia si la VC fes servir un reemplaçament LRU? Perquè?

Volem estudiar la implementació d'aquestes 3 caches com a cache de dades en un processador. Per un programa Pque només fa lectures executat en aquest processador amb memòria ideal (on tots els accessos a memòria tarden 1cicle) 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}).

d) **Calculeu** el ratio *nr* (accessos a memòria per instrucció).

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?

f) Quin es el temps d'execució de P? (en segons)

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?

i) Quin es el temps d'execució de P? (en segons)

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?

k) Quin es el temps d'execució de P? (en segons)

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 de1 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?

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?

o) Quin es el temps d'execució de P? (en segons)

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 HoldingStatus 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 deque 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 ligeramenteinferior de 1,9 GHz

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