

Ejercicio 3.1.

En la fase de diseño de la cache de un procesador con ejecución en orden se están considerando dos alternativas. La primera es utilizar una correspondencia directa, lo que da lugar a una tasa de fallos del 1.4 %, fijando el periodo de reloj en 1 ns. La segunda consiste en utilizar una correspondencia asociativa, lo que permite reducir la tasa de fallos al 1 %, pero el periodo de reloj aumenta hasta 1.25 ns. En caso de acierto, se consume un ciclo de reloj en leer o escribir el dato. Si los programas se ejecutan con un CPI de 2, con 1.5 accesos a memoria por instrucción y la penalización por fallo es de 75 ns, decídase qué alternativa es la mejor.

Solución:

El tiempo de acceso a memoria en un sistema con memorias cache es:

$$T_{acceso} = TA + TF \times PF$$

En nuestro caso:

- Correspondencia directa:

$$T_{acceso} = 1 + 0,014 \times 75 = 2,05 \text{ ns.}$$

- Correspondencia asociativa:

$$T_{acceso} = 1,25 + 0,01 \times 75 = 2 \text{ ns.}$$

Resulta mejor la correspondencia asociativa, pues reduce el tiempo de acceso.

Seguidamente, compararemos ambas alternativas desde el punto de vista del tiempo de ejecución. El tiempo de ejecución de un programa en un computador con memoria cache viene dado por:

$$T_{ejecucion} = I \times CPI \times T + I \times API \times TF \times PF \times T. \text{ Sustituyendo:}$$

- Correspondencia directa:

Como el tiempo de acceso a memoria es de 75 ns y el periodo de reloj es de $T = 1$ ns, la penalización por fallo en ciclos es de $PF = 75$:

$$T_{ejecucion} = I \times 2 \times 1 + I \times 1,5 \times 0,014 \times 75 \times 1 = 3,58I \text{ ns.}$$

- Correspondencia asociativa:

Como el tiempo de acceso a memoria es de 75 ns y el periodo de reloj es de $T = 1,25$ ns, la penalización por fallo en ciclos es de $PF = 60$:

$$T_{ejecucion} = I \times 2 \times 1,25 + I \times 1,5 \times 0,01 \times 60 \times 1,25 = 3,63I \text{ ns.}$$

La correspondencia directa, con menor periodo de reloj, reduce el tiempo de ejecución.

□

Ejercicio 3.2.

Analizar el tiempo de ejecución de un programa en un procesador segmentado con ejecución en orden y CPI=1 bajo las siguientes configuraciones de memoria cache:

1. No emplear memoria cache.
2. Utilizar una cache con una tasa de fallos del 2 %.
3. Utilizar una cache perfecta.

La memoria principal tiene 100 ciclos de reloj de tiempo de acceso y el programa tiene un 50 % de instrucciones de acceso a memoria.

Solución:

El tiempo de acceso a memoria en un sistema con memorias cache es:

$$T_{acceso} = TA + TF \times PF$$

El tiempo de ejecución de un programa en un computador con memoria cache viene dado por:

$$T_{ejecucion} = I \times CPI \times T + I \times API \times TF \times PF \times T.$$

Sustituyendo:

1. No emplear memoria cache.

$$T_{acceso} = 1 + 1(100\%) \times 100 = 101 \text{ ciclos}$$

$$T_{ejecucion} = I \times 1 \times T + I \times 1,5 \times 1(100\%) \times 100 \times T = 151I \text{ ciclos.}$$

2. Utilizar una cache con una tasa de fallos del 2 %.

$$T_{acceso} = 1 + 0,02 \times 100 = 3 \text{ ciclos}$$

$$T_{ejecucion} = I \times 1 \times T + I \times 1,5 \times 0,02 \times 100 \times T = 4I \text{ ciclos.}$$

3. Utilizar una cache perfecta.

$$T_{acceso} = 1 + 0,0 \times 100 = 1 \text{ ciclos}$$

$$T_{ejecucion} = I \times 1 \times T + I \times 1,5 \times 0,0 \times 100 \times T = I \text{ ciclos.}$$

□

Ejercicio 3.3.

Se pretende comparar las prestaciones de dos subsistemas de memoria. El primero tiene sendas memorias cache de instrucciones y datos de 16 KB cada una, mientras que el segundo tiene una única memoria cache de 32 KB. Tras haber ejecutado unos programas de prueba, se ha obtenido que el número de fallos por cada 1000 instrucciones ejecutadas es 3.83 para la cache de instrucciones, 40.9 para la cache de datos y 43.3 para la cache unificada, y que el 36 % de las instrucciones ejecutadas son de acceso a memoria. Si el tiempo de acierto y de fallo es de 1 y 100 ciclos de reloj, respectivamente:

1. Calcular el tiempo de acceso a memoria en ambos esquemas. Considera en dicho cálculo la sobrecarga que puedan inducir los riesgos estructurales que los accesos a memoria puedan provocar.
2. Si el procesador está segmentado, ejecuta las instrucciones en orden con CPI=1 y el programa de prueba se compone de n instrucciones, comparar las dos alternativas en base al tiempo de ejecución.

Solución:

1. Calculo del tiempo de acceso a memoria en ambos esquemas.

El tiempo de acceso a memoria en un sistema con memorias cache es:

$$T_{acceso} = TA + TF \times PF$$

En nuestro caso:

TF (fallos/acceso) = (fallos/instruccion)/(accesos/instruccion). Sustituyendo:

$$TF_I = \frac{3,82/1000}{1} = 0,004 \quad TF_D = \frac{40,9/1000}{0,36} = 0,1144 \quad TF_U = \frac{43,3/1000}{1,36} = 0,0318$$

Caches separadas:

Un programa compuesto por 100 instrucciones realiza por término medio 36 accesos a memoria de datos, con 136 accesos en total. Por lo tanto, la tasa de fallos vendrá dada por:

$$TF_{I+D} = \frac{100}{136} TF_I + \frac{36}{136} TF_D = 0,74 \cdot 0,004 + 0,26 \cdot 0,1144 = 0,0324. \text{ Por lo tanto;}$$

$$T_{acceso} = TA + TF \times PF = 1 + 0,0324 \cdot 100 = 4,24 \text{ ciclos}$$

Cache unificada:

En este caso, hay que tener presente que los accesos a memoria de datos (26 % del total de accesos) van a ocasionar un riesgo estructural que se saldrá con un ciclo de parada:

$$T_{acceso} = TA + TF \times PF = 1 + 0,26 \cdot 1 + 0,0318 \cdot 100 = 4,44 \text{ ciclos}$$

2. Comparación de las dos alternativas en base al tiempo de ejecución.

El tiempo de ejecución de un programa en un computador con memoria cache viene dado por:

$$T_{ejecucion} = I \times CPI \times T + I \times API \times TF \times PF \times T. \text{ Sustituyendo:}$$

Caches separadas:

$$T_{ejecucion} = n \times 1 \times T + n \times 1,36 \times 0,0324 \times 100 \times T = 5,41n \text{ ciclos.}$$

Cache unificada:

$$T_{ejecucion} = n \times (1 + 0,36) \times T + n \times 1,36 \times 0,0318 \times 100 \times T = 5,49n \text{ ciclos.}$$

□

Ejercicio 3.4.

En el diseño de un procesador segmentado con ejecución en orden y $CPI=1$ se pretende evaluar dos alternativas para el sistema de antememorias. La primera basada en una cache de dos niveles. La segunda formada por una cache asociativa por conjuntos de cuatro vías. A continuación se detallan los datos técnicos de ambas alternativas.

Cache multinivel Está compuesta por dos niveles de cache.

Primer nivel: directa Tiempo de acierto, $TA_{L1} = 1$ ciclo; tasa de fallos local, $TF_{L1} = 0,04$.

Segundo nivel: asociativa Tiempo de acierto, $TA_{L2} = 5$ ciclos; tasa de fallos local, $TF_{L2} = 0,50$.

Cache asociativa Una cache asociativa por conjuntos de cuatro vías.

Tiempo de acierto, $TA = 1$ ciclo; tasa de fallos local, $TF = 0,02$.

En ambos casos, la memoria principal tiene un tiempo de acceso de 50 ciclos. También se sabe que el 25 % de las instrucciones ejecutadas son de carga o almacenamiento.

Se solicita:

1. Calcular el tiempo de acceso a memoria en ambos esquemas.
2. Comparar las dos alternativas en base al tiempo de ejecución.

Solución:

Cache multinivel:

El tiempo de acceso de una cache multinivel es:

$$T_{acceso} = TA_{L1} + TF_{L1} \times \underbrace{(TA_{L2} + TF_{L2} \times PF_{L2})}_{PF_{L1}}$$

sustituyendo:

$$T_{acceso} = 1 + 0,04 \times \underbrace{(5 + 0,5 \times 50)}_{PF_{L1}} = 1 + 0,04 \times \underbrace{30}_{PF_{L1}} = 2,2 \text{ ciclos}$$

Utilizando el tiempo de PF_{L1} , el tiempo de ejecución sería:

$$T_{ejecucion} = I \times CPI \times T + I \times API \times TF \times PF \times T$$

$$T_{ejecucion} = I \times (CPI + API \times TF \times PF) \times T$$

sustituyendo:

$$T_{ejecucion} = I \times (1 + 1,25 \times 0,04 \times 30) \times T = I \times 2,5 \times T$$

Cache asociativa:

El tiempo de acceso de una cache convencional es:

$$T_{acceso} = TA_{L1} + TF_{L1} \times PF_{L1}$$

sustituyendo:

$$T_{acceso} = 1 + 0,02 \times 50 = 2 \text{ ciclos}$$

Por otro lado, el tiempo de ejecución sería:

$$T_{ejecucion} = I \times (CPI + API \times TF \times PF) \times T$$

sustituyendo:

$$T_{ejecucion} = I \times (1 + 1,25 \times 0,02 \times 50) \times T = I \times 2,25 \times T$$

Así pues, la segunda alternativa es mejor con una aceleración:

$$S = \frac{T_{multinivel}}{T_{asociativa}} = \frac{I \times 2,5 \times T}{I \times 2,25 \times T} = 1,11$$

□

Ejercicio 3.5.

En el diseño de un computador segmentado con ejecución en orden y $CPI=1$ se dispone de una jerarquía de memoria compuesta por un sistema de antememorias de dos niveles y una memoria principal. Las características son las siguientes:

Primer nivel: correspondencia directa Tiempo de acierto, $TA_{L1} = 1$ ciclo; tasa de fallos local, $TF_{L1} = 0,04$.

Segundo nivel: correspondencia asociativa Tiempo de acierto, $TA_{L2} = 5$ ciclos; tasa de fallos local, $TF_{L2} = 0,50$.

Memoria principal Tiempo de acceso, $TA_{MP} = 20$ ciclos.

Se sabe que el 25 % de las instrucciones ejecutadas son de carga o almacenamiento.

Se propone mejorar el acceso a la memoria principal utilizando la técnica de *palabra crítica cuanto antes* (*critical word first*), que reduce el tiempo de acceso efectivo a la memoria principal a $TA_{MP} = 15$ ciclos. Introducir esta mejora implica aumentar el periodo de reloj en un 10 %.

El aumento en el periodo de reloj nos permite aumentar la asociatividad del segundo nivel de antememorias, reduciendo la tasa de fallos local a $TF_{L2} = 0,40$.

Se solicita cuantificar la mejora obtenida con respecto al tiempo de ejecución.

Solución:

1. El tiempo de acceso de la cache multinivel original es:

$$T_{acceso} = TA_{L1} + TF_{L1} \times \underbrace{(TA_{L2} + TF_{L2} \times PF_{L2})}_{PF_{L1}}$$

$$PF_{L1} = TA_{L2} + TF_{L2} \times PF_{L2} = 5 + 0,5 \times 20 = 5 + 10 = 15$$

El tiempo de ejecución sería:

$$T_{ejecucion} = I \times CPI \times T + I \times API \times TF \times PF \times T$$

$$T_{ejecucion} = I \times (CPI + API \times TF \times PF) \times T$$

sustituyendo:

$$T_{ejecucion} = I \times (1 + 1,25 \times 0,04 \times 15) \times T = I \times 1,75 \times T$$

La penalización por fallo de nivel L1 con la mejora indicada sería:

$$PF'_{L1} = TA'_{L2} + TF'_{L2} \times PF'_{L2} = 5 + 0,4 \times 15 = 5 + 6 = 11$$

Utilizando este tiempo de PF_{L1} , el tiempo de ejecución sería:

$$T'_{ejecucion} = I' \times (CPI' + API' \times TF' \times PF') \times T'$$

sustituyendo:

$$T'_{ejecucion} = I \times (1 + 1,25 \times 0,04 \times 11) \times 1,1 \cdot T$$

$$T'_{ejecucion} = I \times 1,55 \times 1,1 \cdot T = I \times 1,705 \times T$$

Así pues, la mejora obtenida sería:

$$S = \frac{T_{ejecucion}}{T'_{ejecucion}} = \frac{I \times 1,75 \times T}{I \times 1,705 \times T} = 1,026$$

El tiempo de ejecución mejoraría en un 2,6 %.

□

Ejercicio 3.6. Se está diseñando un procesador segmentado de 32 bits con ejecución en orden y cache de instrucciones y de datos separadas. Las características del esquema de memoria son:

- Memoria principal: latencia, 16 *ciclos*; ancho de banda, $W = 2 \text{ bytes/ciclo}$. Con lo que, por ejemplo, el tiempo de transferencia de un bloque de 6 palabras sería: $T_{mp} = 16 + \frac{(6 \times 4)}{2} = 28 \text{ ciclos}$.
- Cache de datos: Tiempo de acierto, $TA^D = 1 \text{ ciclo}$; tasa de fallos, $TF^D = 0,05$.
- Cache de instrucciones: Tiempo de acierto, $TA^I = 1 \text{ ciclo}$; tasa de fallos, $TF^I = 0,02$.
- Tamaño de bloque, $B = 4 \text{ palabras}$.

Se pretenden evaluar dos posibles mejoras para el sistema de caches. A continuación se detallan los datos técnicos de ambas alternativas.

- 1) Incorporar un segundo nivel de cache con: Tiempo de acierto, $TA_{L2} = 6 \text{ ciclos}$; tasa de fallos local, $TF_{L2} = 0,5$. El tamaño de bloque se mantiene.
- 2) Aumentar el tamaño de bloque de ambas caches a 8 *palabras*, duplicando así su tamaño. Este cambio afectará a la tasa de fallos de ambas antememorias: $TF^D = 0,03$ y $TF^I = 0,015$

El CPI, ignorando los fallos de antememoria, es de 1,1 *ciclos/inst* y se sabe que el 25 % de las instrucciones ejecutadas son de carga o almacenamiento. Se solicita analizar la influencia de ambas alternativas sobre el tiempo de ejecución de los programas, indicando cuál de las dos alternativas es la más interesante y cuantificando la mejora obtenida sobre el diseño original.

Solución:

El tiempo necesario para traer un bloque de B palabras de la memoria principal viene dado por la expresión:

$$T_{mp} = 16 + \frac{(B \times 4)}{2} = 16 + B \times 2 \text{ ciclos.}$$

El tiempo de ejecución de teniendo en cuenta la jerarquía de memorias sería:

$$T_{ejec} = I \times CPI \times T + I \times API \times TF \times PF \times T$$

Cuando se tiene una arquitectura con antememoria de datos y de instrucciones el numero de accesos por instrucción y la tasa de fallos se pueden separar:

$$T_{ejec} = I \times CPI \times T + I \times ((API^I \times TF^I + API^D \times TF^D) \times PF) \times T$$

Aplicándolo al diseño original del subsistema de memoria:

$$CPI = 1,1, API^I = 1, TF^I = 0,02, API^D = 0,25, TF^D = 0,05 \text{ y } PF = 16 + 4 \times 2 = 24 \text{ ciclos.}$$

Sustituyendo:

$$T_{ejec} = I \times 1,1 \times T + I \times ((1 \times 0,02 + 0,25 \times 0,05) \times 24) \times T = I \times 1,88 \times T$$

Veamos ahora el tiempo de ejecución con cada una de las mejoras propuestas:

- Dos niveles de cache.

En este caso, la penalización en caso de fallo queda:

$$PF = TA_{L2} + TF_{L2} \times T_{mp} = 6 + 0,5 \times 24 = 18 \text{ ciclos.}$$

Sustituyendo en la expresión del tiempo de ejecución:

$$T_{ejec} = I \times 1,1 \times T + I \times ((1 \times 0,02 + 0,25 \times 0,05) \times 18) \times T = I \times 1,69 \times T$$

- Aumentar el tamaño de bloque.

En este caso, además de las tasas de fallo, se modifica la penalización en caso de fallo, al cambiar el tamaño de bloque:

$$PF = 16 + 8 \times 2 = 32 \text{ ciclos.}$$

Sustituyendo en la expresión del tiempo de ejecución:

$$T_{ejec} = I \times 1,1 \times T + I \times ((1 \times 0,015 + 0,25 \times 0,03) \times 32) \times T = I \times 1,82 \times T$$

Claramente, la primera opción es la mejor. La aceleración obtenida respecto del diseño original es $\frac{I \times 1,88 \times T}{I \times 1,69 \times T} = 1,1124$. Por lo tanto, utilizar dos niveles de cache mejora el tiempo de ejecución en un 11,24 %.

□

Ejercicio 3.7.

Un computador dispone de un procesador semejante al MIPS, ejecución en orden de instrucciones, con un ancho de palabra de 32 bits y que trabaja a una frecuencia de 100 MHz, con un CPI = 1.3. El 20 % de las instrucciones que ejecuta son lecturas de memoria, y el 10 % son escrituras.

La memoria cache es de un nivel, separada en 16 KB para instrucciones y 16 KB para datos. El tamaño de bloque es de 16 bytes y la correspondencia es asociativa por conjuntos de dos. Sigue las políticas *write-through* y *no-write allocate*. La tasa de fallos es de 2 % para las lecturas y 5 % para las escrituras.

Para las escrituras, el procesador dispone de buffers de escrituras.

La memoria principal ofrece una latencia de lectura de 40 ns. Una vez cumplida esta latencia, suministra las palabras en intervalos de 10 ns. Así, la primera palabra llega a los 50ns, la segunda a los 60ns, y así consecutivamente. Inicialmente, la carga de un bloque en la cache es convencional (esto es, no aplica *Critical Word First*—palabra crítica cuanto antes ni *Early Restart*—continuar cuanto antes)

Calcula:

1. La penalización por fallo de lectura, expresada en ciclos de reloj y en segundos, cuando se aplica carga convencional.
2. La penalización por fallo de lectura, expresada en ciclos de reloj y en segundos, cuando se aplica carga *Critical Word First*.
3. La penalización por fallo de escritura, expresada en ciclos de reloj y en segundos
4. El tiempo de ejecución de un programa de 10 millones de instrucciones, suponiendo que se aplica carga convencional.

Solución:

1. Penalización por fallo de lectura con carga convencional:

Si el ancho de palabra es de 32 bits y el tamaño de bloque de 16 bytes, las ráfagas para cargar un bloque en la cache son de 4 palabras. El tiempo necesario es $PF = (5 + 1 + 1 + 1) \text{ ciclos} = 8 \times 10 \text{ ns} = 80 \text{ ns}$

2. Penalización por fallo de lectura con carga *Critical Word First*:

El tiempo necesario es $PF = 5 \text{ ciclos} = 5 \times 10 \text{ ns} = 50 \text{ ns}$

3. La penalización por fallo de escritura: ninguna, pues con la política *no-write allocate* y los buffers de escritura, el procesador nunca ha de esperar para transferencias con memoria

4. El tiempo de ejecución es $T = I \cdot CPI \cdot t_C + I \cdot LPI \cdot TFL \cdot PFL = 10 \cdot 10^6 \cdot 1,3 \cdot 10ns + 10 \cdot 10^6 \cdot 1,2 \cdot 0,02 \cdot 80ns$

□

Ejercicio 3.8. Un computador tiene un procesador segmentado compatible binario con el MIPS y que además incorpora instrucciones aritméticas que operan con datos en memoria (modelo de ejecución registro-memoria). Ejecuta las instrucciones en orden, trabaja a una frecuencia de 100 MHz y tiene una jerarquía de memoria con caches de instrucciones y de datos separadas, cada una con un solo puerto. El 15 % de las instrucciones que ejecuta son de carga, el 5 % son de almacenamiento y el 40 % son instrucciones aritméticas, de las cuales el 25 % tienen un operando en memoria.

Las etapas que atraviesan las instrucciones son: IF (búsqueda de la instrucción), ID (decodificación y lectura de operandos en registros), ME1 (lectura de los operandos en memoria, en su caso), EX (ejecución operaciones aritméticas), ME2 (lectura/escritura en memoria) y WB (escritura en el banco de registros). El 10 % de las instrucciones de carga sufren un riesgo estructural en el acceso a la memoria de datos (ME2) con una instrucción aritmética con un operando en memoria (ME1). Salvo por esos riesgos, e ignorando los fallos de antememoria, el CPI alcanzado por el procesador es 1.

La tasa de fallos de las antememorias es de 2 % para la cache de instrucciones y 5 % para la cache de datos. La memoria principal tiene una penalización de 40 ciclos.

Se solicita:

1. Calcular el tiempo de ejecución de un programa P, que ejecuta 2000 millones de instrucciones, sin tener en cuenta los fallos de cache.
2. Calcular el número medio de accesos por instrucción (API) a la memoria de instrucciones y a la memoria de datos.
3. Calcular el tiempo de ejecución real del programa P.

Solución:

1. Tiempo de ejecución ideal.

Hay que tener en cuenta que el 10 % de las instrucciones de carga (15 % del total) sufren un riesgo estructural que obliga a insertar un ciclo de parada:

$$CPI = 1 + 0,15 \cdot 0,1 \cdot 1 = 1,015$$

$$T = 10 \text{ ns.}$$

$$T_{ej} = I \cdot CPI \cdot T = (2000 \cdot 10^6) \cdot 1,015 \cdot 10^{-9} = 20,3 \text{ s.}$$

2. Cálculo de API .

Todas las instrucciones necesitan acceder a la memoria de instrucciones para buscarse:

$$API_i = 1 \text{ acceso/instrucción}$$

Las instrucciones de carga (15 %), las de almacenamiento (5 %) y el 25 % de las aritméticas (40 %) necesitan, además, acceder a la memoria de datos:

$$API_d = 0,15 + 0,05 + (0,25 \cdot 0,4) = 0,3 \text{ accesos/instrucción}$$

$$API = API_i + API_d = 1,3 \text{ accesos/instrucción}$$

3. Tiempo de ejecución real.

$$T_{ej} = T_{ej_{ideal}} + T_{memoria}$$

Un programa compuesto por 100 instrucciones realiza por término medio 30 accesos a memoria de datos, con 130 accesos en total. Por lo tanto, la tasa de fallos combinada vendrá dada por:

$$TF_{I+D} = \frac{100}{130} TF_I + \frac{30}{130} TF_D = 0,77 \cdot 0,02 + 0,23 \cdot 0,05 = 0,0269.$$

$$T_{memoria} = I \cdot API \cdot TF \cdot PF \cdot T = (2000 \cdot 10^6) \cdot 1,3 \cdot 0,0269 \cdot 40 \cdot 10^{-9} = 27,98 \text{ s}$$

Sustituyendo:

$$T_{ej} = 20,3 + 27,98 = 48,28 \text{ s}$$

Otra opción equivalente sería modificar la ecuación del $T_{memoria}$ para que refleje la existencia de antememoria de instrucciones y datos separadas:

$$T_{memoria} = I \cdot ((API_i \cdot TF_i) + (API_d \cdot TF_d)) \cdot PF \cdot T = (2000 \cdot 10^6) \cdot ((1 \cdot 0,02) + (0,3 \cdot 0,05)) \cdot 40 \cdot 10^{-9} = 28 \text{ s.}$$

Ejercicio 3.9.

Un computador tiene los siguientes componentes:

- Un procesador *load/store* semejante al MIPS, con ejecución en orden y un ancho de palabra 64 bits y que trabaja a una frecuencia de 100 MHz, con un CPI = 1.3. El 20 % de las instrucciones que ejecuta son lecturas de memoria, y el 10 % son escrituras.
- La memoria cache es de un nivel, separada en 16 KB para instrucciones y 16 KB para datos. El tamaño de bloque es de 32 bytes y la correspondencia es asociativa por conjuntos de dos. Sigue las políticas *write-through* y *no-write allocate*. La tasa de fallos es de 2 % para las lecturas y 5 % para las escrituras.
- Para las escrituras, el procesador dispone de buffers.
- La memoria principal es SDR SDRAM y su temporización en ciclos se describiría como $CL - t_{RCD} - t_{RP} = 1 - 2 - 2$. Se ha medido que en el 90 % de los casos, los accesos encuentran abierta la fila objetivo. La memoria trabaja a 100 MHz. El ancho de bus es de 64 bits.

Calcula:

1. La penalización por fallo de lectura, expresada en ciclos de reloj y en segundos. con carga de bloque convencional (esto es, sin aplicar *Critical Word First* ni *Early Restart*—continuar cuanto antes) .
2. La penalización por fallo de lectura, expresada en ciclos de reloj y en segundos, cuando se aplica carga *Critical Word First*.
3. La penalización por fallo de escritura, expresada en ciclos de reloj y en segundos
4. El tiempo de ejecución de un programa de 10 millones de instrucciones, en dos casos:
 - Con carga convencional del bloque
 - Cuando se aplica *Critical Word First*

Solución:

Si los bloques son de 32 bytes y el ancho de bus de 64 bits, por tanto $B = 32$ bytes y $Bw = 8$ bytes/ciclo.

1. En el cálculo de la penalización PFL por fallo de lectura, hay que considerar dos casos:

- Cuando el acceso encuentra abierta la fila objetivo,

$$PFL \text{ (fila abierta)} = CL + \frac{B}{B_w} = 5 \text{ ciclos} = 50 \text{ ns}$$

- Cuando el acceso encuentra cerrada la fila objetivo, la penalización será:

$$PFL \text{ (fila cerrada)} = t_{RP} + t_{RCD} + CL + \frac{B}{B_w} = 9 \text{ ciclos} = 90 \text{ ns}$$

- El promedio será de

$$PFL \text{ (promedio)} = 0,9 \times 5 + 0,1 \times 9 = 5,4 \text{ ciclos} = 54 \text{ ns}$$

2. La penalización por fallo de lectura con *Critical Word First* es idéntica a la de un bloque de sólo una palabra ($B = 8$ bytes). Por tanto:

- Cuando el acceso encuentra abierta la fila objetivo, la penalización será

$$PFL \text{ (fila abierta)} = CL + \frac{B}{B_w} = 2 \text{ ciclos} = 20 \text{ ns}$$

- Cuando el acceso encuentra cerrada la fila objetivo, la penalización será:

$$PFL \text{ (fila cerrada)} = t_{RP} + t_{RCD} + CL + \frac{B}{B_w} = 6 \text{ ciclos} = 60 \text{ ns}$$

- El promedio será de

$$PFL \text{ (promedio)} = 0,9 \times 2 + 0,1 \times 6 = 2,4 \text{ ciclos} = 24 \text{ ns}$$

3. No hay ninguna penalización por fallo de escritura, pues con la política *no-write allocate* y los buffers de escritura, el procesador nunca ha de esperar para transferencias con memoria.
4. El tiempo de ejecución del programa de 10 millones de instrucciones será

$$t = I \times CPI \times T + I \times LPI \times TFL \times PFL$$

- Con carga convencional del bloque:

$$\begin{aligned} T &= 10 \cdot 10^6 \text{ instrucciones} \times 1,3 \text{ ciclos/instrucción} \times 10 \cdot 10^{-9} \text{ s/ciclo} + \\ &+ 10 \cdot 10^6 \text{ instrucciones} \times 1,2 \text{ lecturas/instrucción} \times \\ &\quad \times 0,02 \text{ fallos/lectura} \times 54 \cdot 10^{-9} \text{ s/fallo} \\ &\simeq 130 + 13 = 143 \text{ ms} \end{aligned}$$

- Cuando se aplica *Critical Word First* sólo cambia la penalización por fallo, y resulta:

$$T = 130 + 10 \cdot 10^6 \times 1,2 \times 0,02 \times 24 \simeq 130 + 6 = 136 \text{ ms}$$

□

Ejercicio 3.10.

Sea un procesador MIPS superescalar con ejecución fuera de orden y tamaño de palabra de 64 bits, funcionando a una frecuencia de reloj de 3.6GHz.

Para una carga típica se obtiene un CPI en ausencia de fallos de 0.4 ciclos. Dicha carga tiene un 25 % de instrucciones de tipo *load* y un 10 % de instrucciones de tipo *store*.

El procesador dispone de dos niveles de memoria *cache*, con caches separadas en el primer nivel (L1I, L1D y L2). Las tasas de fallos locales para la carga estudiada son $TF_{L1I} = 0,5 \%$, $TF_{L1D} = 2,6 \%$ (L1D lectura), $TF_{E_{L1D}} = 6,5 \%$ (L1D escritura) y $TF_{L2} = 36 \%$. El tiempo de acierto es de 1 ciclo para la cache L1 y 8 ciclos para la cache L2. Ambos niveles trabajan con un tamaño de bloque de 128 bytes. El procesador dispone de buffers de escritura. La ejecución fuera de orden permite ocultar el 30 % de la penalización por fallo de la cache L1 de datos, siendo el resto de solapamientos despreciable.

Por otra parte, la memoria principal empleada es una SDRAM DDR4 2400MHz, con un tiempo promedio de lectura de bloque de 38,6 ciclos de bus de memoria.

Calcula:

1. La penalización de fallo para la *cache* L1.
2. El tiempo de ejecución de un programa formado por 15 millones de instrucciones.
3. Existe la posibilidad de realizar una nueva implementación del procesador duplicando el tamaño de las *cache* L1, lo que reduciría la tasa de fallos a $TF_{L1I} = 0,3 \%$, $TF_{L1D} = 1,5 \%$ (L1D lectura), $TF_{E_{L1D}} = 4,5 \%$ (L1D escritura). No obstante esta memoria sería más lenta que la original lo que ha condicionado al equipo de diseño a reducir la frecuencia del procesador a 2.8Ghz. Por su parte la cache L2 aumentaría su tasa de fallos a 57 % y su tiempo de acierto pasaría de 8 ciclos a 6 ciclos. Calcular si esta nueva configuración ofrece una mejora y en tal caso cuantificarla.
4. En el apartado anterior la inclusión de una *cache* L1 de mayor tamaño ha tenido una serie de consecuencias. Intenta dar una posible explicación a cada una de las siguientes modificaciones:
 - a) Se ha tenido que reducir la frecuencia de reloj del procesador.

- b) Se han reducido los ciclos del tiempo de acierto de la cache L2.

Solución:

1. La penalización de fallo para la *cache* L1.

$$PF_{L1} = T_{L2} = TA_{L2} + TF_{L2} \times PF_{L2}$$

Sabiendo que $PF_{L2} = T_{MP}$ (tiempo de lectura de bloque en memoria principal), que nos dan en ciclos del bus de memoria y que hemos de pasar a *ns*,

$$PF_{L2} = T_{MP} = 38,6 \cdot \frac{1}{2,4 \cdot 10^9} = 16,08 \cdot 10^{-9} s \equiv 16,08 \text{ ns}$$

la tasa de fallo de L2 nos la dan como dato,

$$TF_{L2} = 0,36$$

tiempo de acierto de la cache de segundo nivel (nos lo dan en ciclos del procesador) que pasamos a *ns*,

$$TA_{L2} = 8 \times \frac{1}{3,6 \cdot 10^9} = 2,22 \text{ ns}$$

finalmente,

$$PF_{L1} = T_{L2} = 2,22 \text{ ns} + 0,36 \times 16,08 \text{ ns} = 8 \text{ ns}$$

2. El tiempo de ejecución de un programa formado por 15 millones de instrucciones.

Teniendo en cuenta que ejecuta las instrucciones fuera de orden y tiene buffers de escritura, el tiempo de ejecución se calcularía como (el término T_{mem} no aparece porque la penalización ya está en segundos),

$$T_{ex} = I \times CPI \times T_{CPU} + I \times TF_{L1I} \times PF_{L1I} + I \times API_{Load} \times TFL_{L1D} \times FNS \times PFL_{L1D}$$

Siendo $API_{Load} = 0,25$, $PFL_{L1D} = PF_{L1I} \rightarrow PF_{L1}$, $FNS = 1 - 0,3 = 0,7$ y el resto de datos nos los dan o ya están calculados.

$$T_{ex} = 15 \cdot 10^6 \times 0,4 \times \frac{1}{3,6 \times 10^9} + 15 \cdot 10^6 \times 0,005 \times 8 \cdot 10^{-9} + 15 \cdot 10^6 \times 0,25 \times 0,026 \times 0,7 \times 8 \cdot 10^{-9} = 2,81 \text{ ms}$$

3. Existe la posibilidad de realizar una nueva implementación del procesador duplicando el tamaño de las *cache* L1, lo que reduciría la tasa de fallos a $TF_{L1I} = 0,3\%$, $TFL_{L1D} = 1,5\%$ (L1D lectura), $TFE_{L1D} = 4,5\%$ (L1D escritura). No obstante esta memoria sería más lenta que la original lo que ha condicionado al equipo de diseño a reducir la frecuencia del procesador a 2.8Ghz. Por su parte la cache L2 aumentaría su tasa de fallos a 57 % y su tiempo de acierto pasaría de 8 ciclos a 6 ciclos. Calcular si esta nueva configuración ofrece una mejora y en tal caso cuantificarla.

En este caso habrá que recalcular PF_{L1} ya que depende del TA_{L2} y del periodo del reloj del procesador que se han modificado,

$$PF_{L1} = T_{L2} = 6 \times \frac{1}{2,8 \text{ GHz}} + 0,57 \times 16,08 \text{ ns} = 11,31 \text{ ns}$$

Sustituyendo en la ecuación del tiempo de ejecución, junto con TF_{L1} y T_{CPU} que han sido también modificados,

$$T_{ex} = 15 \cdot 10^6 \times 0,4 \times \frac{1}{2,8 \times 10^9} + 15 \cdot 10^6 \times 0,003 \times 11,31 \cdot 10^{-9} + 15 \cdot 10^6 \times 0,25 \times 0,015 \times 0,7 \times 11,31 \cdot 10^{-9} = 3,1 \text{ ms}$$

Comparando ambas ejecuciones,

$$A = \frac{3,1 \text{ ms}}{2,81 \text{ ms}} = 1,103 \rightarrow 10,3 \%$$

comprobamos que la configuración inicial es 1.103 veces (10.3 %) mejor que la modificada.

4. En el apartado anterior la inclusión de una *cache* L1 de mayor tamaño ha tenido una serie de consecuencias. Intenta dar una posible explicación a cada una de las siguientes modificaciones:

- a) Se ha tenido que reducir la frecuencia de reloj del procesador.

En caso de acierto la memoria L1 debe poder suministrar el dato en un ciclo del procesador, si introducimos una memoria L1 más lenta esto podría ya no ser posible. Por ello tenemos que aumentar la duración de un ciclo del procesador, y por lo tanto reducir su frecuencia.

- b) Se han reducido los ciclos del tiempo de acierto de la L2.

La L2 mantiene sus características, es igual de rápida, si se ha reducido el número de ciclos es porque éstos tienen ahora mayor duración.

$$8 \times \frac{1}{3,6 \cdot 10^{-9}} \sim 6 \times \frac{1}{2,8 \cdot 10^{-9}}$$

□

Ejercicio 3.11.

Un sistema tiene instalada una memoria Samsung Hynix Micron DDR2 PC2-6400 con temporización $CL-t_{RCD}-t_{RP}$ 10-10-10 y ancho de bus de 64 bits. En cada acceso, la memoria transfiere un bloque de memoria de 64 bytes. De media, la probabilidad de que un acceso a memoria encuentre abierta la fila donde se encuentra el bloque es del 40 %.

Responde, justificando las respuestas:

1. ¿Cuál es el tiempo de lectura de un bloque en memoria principal? Calcúlalo en ciclos de bus de memoria.
2. Se han considerado tres opciones para mejorar el tiempo de lectura:
 - a) Reducir a la mitad el tiempo entre el envío de la dirección de columna por el bus de memoria y el comienzo de la recepción del bloque.
 - b) Reducir a la mitad la latencia del comienzo de la recepción de los datos cuando el acceso encuentra abierta otra fila diferente de la que contiene el bloque.
 - c) Duplicar el ancho de bus de memoria.

Para cada una de las tres opciones, calcula el tiempo de lectura en ciclos de bus.

3. Calcula el tiempo de lectura en ciclos de bus para la memoria original aplicando la técnica *Early Restart*. Asume que cada palabra de 64 bits del bloque de cache tiene la misma probabilidad de ser la requerida por el procesador.

Solución:

1. ¿Cuál es el tiempo de lectura de un bloque en memoria principal? Calcúlalo en ciclos de bus de memoria.

El tiempo de acceso a memoria en ciclos sigue la siguiente ecuación:

$$TA_{MM} = L \times (1 - TA_{BF}) + Lr \times TA_{BF} + \frac{B}{BW},$$

donde $L = 10 + 10 + 10 = 30$, $Lr = 10$, $TA_{BF} = 0,40$, $B = 64 \text{ bytes}$ y $BW = 16$ (DDR), por tanto:

$$TA_{MM} = 30 \times (1 - 0,40) + 10 \times 0,40 + \frac{64}{16} = 26 \text{ ciclos.}$$

2. Para cada una de las tres opciones, calcula el tiempo de lectura en ciclos de bus.

- a) Reducir a la mitad el tiempo entre el envío de la dirección de columna por el bus de memoria y el comienzo de la recepción del bloque.

Esta opción implica que $Lr = CL = 10/2 = 5$ y $L = 10/2 + 10 + 10 = 25$, por lo tanto:

$$TA_{MM} = 25 \times (1 - 0,40) + 5 \times 0,40 + \frac{64}{16} = 21 \text{ ciclos.}$$

- b) Reducir a la mitad la latencia del comienzo de la recepción de los datos cuando el acceso encuentra abierta otra fila diferente de la que contiene el bloque.

Esta opción implica que $L = 30/2 = 15$, por lo tanto:

$$TA_{MM} = 15 \times (1 - 0,40) + 10 \times 0,40 + \frac{64}{16} = 17 \text{ ciclos.}$$

- c) Duplicar el ancho de bus de memoria.

Esta opción implica que $BW = 16 \times 2 = 32$, por lo tanto:

$$TA_{MM} = 30 \times (1 - 0,40) + 10 \times 0,40 + \frac{64}{32} = 24 \text{ ciclos.}$$

3. Calcula el tiempo de lectura en ciclos de bus para la memoria original aplicando la técnica *Early Restart*. Asume que cada palabra de 64 bits del bloque de cache tiene la misma probabilidad de ser la requerida por el procesador.

Mediante la técnica *Early Restart*, se permite al procesador continuar inmediatamente después de que la palabra requerida se recibe por el bus de memoria.

Como la memoria es DDR, la primera palabra se recibe 1 semiciclo después de que comience la recepción del bloque, la segunda tras 2 semiciclos y así sucesivamente.

La probabilidad de que una palabra de un bloque sea la requerida por el procesador se reparte equitativamente entre las 8 palabras del bloque.

Por lo tanto, el tiempo medio de acceso a memoria queda como sigue:

$$TA_{MM} = 30 \times (1 - 0,40) + 10 \times 0,40 + \frac{8/8 + 16/8 + 24/8 + 32/8 + 40/8 + 48/8 + 56/8 + 64/8}{16} = 24,25 \text{ ciclos.}$$

□

Ejercicio 3.12. Un computador tiene los siguientes componentes:

- Un procesador *load/store* semejante al MIPS, ejecución fuera de orden, un ancho de palabra de 64 bits y que trabaja a una frecuencia de reloj de 2,5 GHz. Se ha medido que el valor medio del CPI es 1,5, y que el 20 % de las instrucciones que ejecuta son lecturas de memoria, y el 5 % son escrituras. El procesador también dispone de buffers de escritura.
- Una memoria cache de dos niveles. El primer nivel L1 tiene cache de instrucciones y de datos separadas. El tiempo de acierto es de 1 ciclo para las caches L1 y de 5 ciclos para la cache L2. Las tasas de fallos locales son del 1 % para la cache L1 de instrucciones, 10 % para las lecturas en la cache L1 de datos, 5 % para las escrituras en la cache L1 de datos y 40 % para la cache L2. El tamaño de bloque es de 128 Bytes en ambos niveles.
- La memoria principal es SDRAM DDR4 y sus parámetros temporales son $CL - t_{RCD} - t_{RP} = 10 - 11 - 12$ a una frecuencia de reloj de 800 MHz.

Calcula, justificando las respuestas:

1. El tiempo necesario para traer un bloque desde la memoria principal a la cache L2 cuando la fila objetivo está abierta, expresado en ciclos de reloj de memoria y en nanosegundos.
2. El tiempo necesario para traer un bloque desde la memoria principal a la cache L2 cuando la fila objetivo está cerrada, expresado en ciclos de reloj de memoria y en nanosegundos.
3. El tiempo de ejecución de un *benchmark* con 10 millones de instrucciones, suponiendo que la cache L1 es perfecta (tasa de fallos igual a cero).
4. El tiempo de ejecución extra, debido a fallos en la jerarquía de cache, para un *benchmark* con 10 millones de instrucciones, suponiendo que el tiempo medio de acceso a memoria (penalización por fallo de la cache L2) es de 40 ns. La ejecución fuera de orden permite ocultar el 70 % de la penalización por fallo de la cache L1 de datos, siendo el resto de solapamientos despreciable.
5. El tiempo medio de acceso a la cache L1 de datos, suponiendo que el tiempo medio de acceso a memoria (penalización por fallo de la cache L2) es de 40 ns.
6. La tasa de fallos combinada del nivel L1 de cache.
7. La tasa de fallos global de la jerarquía de cache.

Solución:

1. El tiempo necesario para traer un bloque desde la memoria principal a la cache L2 cuando la fila objetivo está abierta, expresado en ciclos de reloj de memoria y en nanosegundos.

Es igual al tiempo de acceso a memoria cuando $L = CL$,

$$MP_{L2} \text{ (en ciclos)} = L + \frac{1}{B_w} \cdot B$$

El tamaño de bloque es de 128 bytes ($B = 128$). En las memorias DDR SDRAM el ancho de bus es de 64 bits (8 bytes) y se realizan 2 transferencias por ciclo, por lo tanto $B_w = 16$ bytes/ciclo.

$$L = CL \text{ ciclos} = 10 \text{ ciclos}$$

Así,

$$MP_{L2} \text{ (en ciclos)} = 10 + \frac{1}{16} \cdot 128 = 18 \text{ ciclos}$$

$$MP_{L2} \text{ (en ns)} = 18 \text{ ciclos} \cdot \frac{1}{0,8 \text{ ciclos/ns}} = 22,5 \text{ ns}$$

2. El tiempo necesario para traer un bloque desde la memoria principal a la cache L2 cuando la fila objetivo está cerrada, expresado en ciclos de reloj de memoria y en nanosegundos.

Es igual al tiempo de acceso a memoria cuando $L = CL + t_{RCD} + t_{RP}$,

$$MP_{L2} \text{ (en ciclos)} = L + \frac{1}{B_w} \cdot B$$

$$L = CL + t_{RCD} + t_{RP} \text{ ciclos} = 33 \text{ ciclos}$$

Así,

$$MP_{L2} \text{ (en ciclos)} = 33 + \frac{1}{16} \cdot 128 = 41 \text{ ciclos}$$

$$MP_{L2} \text{ (en ns)} = 41 \text{ ciclos} \cdot \frac{1}{0,8 \text{ ciclos/ns}} = 51,25 \text{ ns}$$

3. El tiempo de ejecución de un *benchmark* con 10 millones de instrucciones, suponiendo que la cache L1 es perfecta (tasa de fallos igual a cero).

$$T_{ex} = I \times CPI \times T + I \times API \times TF \times PF$$

Ya que la tasa de fallos (TF) es 0:

$$\begin{aligned} T_{ex} &= 10 \cdot 10^6 \text{ instrucciones} \times 1,5 \text{ ciclos/instruccion} \times 0,4 \cdot 10^{-9} \text{ s/ciclo} \\ &= 6 \text{ ms} \end{aligned}$$

4. El tiempo de ejecución extra, debido a fallos en la jerarquía de cache, para un *benchmark* con 10 millones de instrucciones, suponiendo que el tiempo medio de acceso a memoria (penalización por fallo de la cache L2) es de 40 ns. La ejecución fuera de orden permite ocultar el 70 % de la penalización por fallo de la cache L1 de datos, siendo el resto de solapamientos despreciable.

La penalización por fallo de la L1 es,

$$PF_{L1} = TA_{L2} + TF_{L2} \times PF_{L2} = 5 \text{ ciclos} \times 0,4 \text{ ns/ciclo} + 0,4 \times 40 \text{ ns} = 18 \text{ ns}$$

Teniendo en cuenta que ejecuta las instrucciones fuera de orden, tiene buffers de escritura, y que la $PF_{L1I} = PFL_{L1D} \rightarrow PF_{L1}$, el tiempo extra de ejecución se calcularía como (el término T_{mem} no aparece porque la penalización ya está en segundos),

$$T_{extra} = I \times API_I \times TF_{L1I} \times PF_{L1} + I \times API_{Load} \times TFL_{L1D} \times FNS \times PF_{L1}$$

$$T_{extra} = 10 \cdot 10^6 \times 1 \times 0,01 \times 18 \cdot 10^{-9} + 10 \cdot 10^6 \times 0,2 \times 0,1 \times 0,3 \times 18 \cdot 10^{-9} = 2,88 \text{ ms}$$

5. El tiempo medio de acceso a la cache L1 de datos, suponiendo que el tiempo medio de acceso a memoria (penalización por fallo de la cache L2) es de 40 ns.

El tiempo medio de acceso a la L1 de datos, teniendo en cuenta que los fallos de escritura no producen penalización (hay buffers de escritura) es,

$$T_{L1D} = TA_{L1} + TF_{L1D} \times PF_{L1} = TA_{L1} + TF_{L1D} \times (TA_{L2} + TF_{L2} \times PF_{L2})$$

$$T_{L1D} = 1 \text{ ciclos} \times 0,4 \text{ ns/ciclo} + 0,1 \times (5 \text{ ciclos} \times 0,4 \text{ ns/ciclos} + 0,4 \times 40 \text{ ns}) = 2,2 \text{ ns}$$

6. La tasa de fallos combinada del nivel L1 de cache.

$$TF'_{L1} = TF_{L1I} \frac{API_I}{API} + TFL_{L1D} \frac{API_{Load}}{API} + TFE_{L1D} \frac{API_{Store}}{API}$$

$$TF'_{L1} = 0,01 \frac{1}{1,25} + 0,1 \frac{0,2}{1,25} + 0,05 \frac{0,05}{1,25} = 0,026 \rightarrow 2,6 \%$$

7. La tasa de fallos global de la jerarquía de cache.

$$TF_{global} = TF'_{L1} \times TF_{L2} = 0,026 \times 0,4 = 0,0104 \rightarrow 1,04 \%$$

□

Ejercicio 3.13.

El siguiente diagrama i-t muestra la ejecución de un fragmento de bucle que ha sido optimizado con la técnica de *loop unrolling*:

		1	2	3	4	5	6	7	8	9	10	11	...	39	40	41	42	43	44	45	46	47	48	49	50		
loop:	l.d f1, 0(r1)	IF	I	AC	L1	L2	L3	L4	L5	L6	L7	L8						L36	L37	WB	C						
	l.d f2, 8(r1)		IF	I	AC													L1	WB	C							
	l.d f3, 16(r1)			IF	I	AC													L1	WB	C						
	l.d f4, 24(r1)				IF	I	AC												L1	WB	C						
	mul.d f1, f0, f1					IF	I												M1	M2	M3	WB	C				
	mul.d f2, f0, f2						IF	I											M1	M2	M3	WB	C				
	mul.d f3, f0, f3							IF	I											M1	M2	M3	WB	C			
	mul.d f4, f0, f4								IF	I											M1	M2	M3	WB	C		
	.																										
	.																										
	.																										

El procesador tiene soporte *hardware* para ejecución fuera de orden y especulación, puede lanzar una instrucción por ciclo de reloj y dispone de un operador de carga no segmentado y uno de multiplicación segmentado para operandos de coma flotante. Dispone de un único nivel de cache L1. La memoria principal está basada en tecnología DDR3 y su frecuencia de reloj coincide con la del núcleo del procesador.

Se pide:

1. Justificar el diferente comportamiento que tienen la primera y la segunda instrucciones de carga.
2. ¿Cuál es el tiempo de acceso (en ciclos de reloj) a la cache de datos L1 en caso de acierto (sin incluir el retardo de la etapa de calculo de la dirección de memoria, etapa AC)?
3. ¿Cuál es la penalización por fallo?
4. En la segunda iteración del bucle (no mostrada en el diagrama), la primera instrucción de carga (l.d f1,0(r1)) reduce el tiempo para ejecutarse completamente de 42 a 18 ciclos. ¿A qué se debe ese comportamiento? El resto de instrucciones consumen el mismo número de ciclos que los mostrados en el diagrama.
¿Cuál es el valor de $t_{RP} + t_{RCD}$ para los módulos DIMM de memoria?
5. ¿Cuál es el tamaño de bloque de cache en bytes?

Solución:

1. Justificar el diferente comportamiento que tienen la primera y la segunda instrucciones de carga.
La primera carga falla en el caché de datos de primer nivel y tiene que acceder al siguiente nivel en la jerarquía hasta que encuentre los datos. Luego, trae un bloque de caché a la memoria caché de datos de primer nivel. La segunda carga accede a la siguiente palabra de 64 bits en el mismo bloque de caché. Por lo tanto, acierta en el caché de primer nivel y la latencia de acceso es mucho más corta.
2. ¿Cuál es el tiempo de acceso (en ciclos de reloj) a la cache de datos L1 en caso de acierto (sin incluir el retardo de la etapa de calculo de la dirección de memoria, etapa AC)?
Desde la segunda y siguientes instrucciones de carga, se necesita un ciclo para acceder al caché de datos de primer nivel.
3. ¿Cuál es la penalización por fallo?
La primera carga requiere 37 ciclos para acceder a la memoria caché de datos de primer nivel y luego acceder a la memoria principal. Como el tiempo de acceso a la memoria caché es de un ciclo, se necesitan 36 ciclos para acceder a la memoria y llevar el bloque correspondiente a la memoria caché.
4. En la segunda iteración del bucle (no mostrada en el diagrama), la primera instrucción de carga (l.d f1,0(r1)) reduce el tiempo para ejecutarse completamente de 42 a 18 ciclos. ¿A qué se debe ese comportamiento? El resto de instrucciones consumen el mismo número de ciclos que los mostrados en el diagrama. ¿Cuál es el valor de $t_{RP} + t_{RCD}$ para los módulos DIMM de memoria?

En primer lugar, tenga en cuenta que el predictor de saltos predecirá incorrectamente la condición de salto en la primera iteración, por lo que tendrá que vaciar el ROB e iniciar de nuevo la búsqueda de instrucciones. Por lo tanto, la razón de la menor cantidad de ciclos se tiene que encontrar fuera del procesador.

Antes de ejecutar la primera iteración de bucle, el búfer de fila correspondiente en la memoria principal tenía la fila incorrecta abierta. Por lo tanto, la primera carga en la primera iteración tuvo que cerrar la fila incorrecta, abrir la correcta y acceder al bloque solicitado.

La primera carga en la segunda iteración falla en la memoria caché y tiene que acceder a la memoria principal. Sin embargo, esta vez, la fila correcta está abierta y la penalización por fallo es mucho más corta (12 ciclos en lugar de 36). La diferencia entre ambos accesos es $t_{RP} + t_{RCD}$. Por tanto, $t_{RP} + t_{RCD}$ es 24 ciclos.

5. ¿Cuál es el tamaño de bloque de cache en bytes?

Como la primera carga en la segunda iteración falla en la caché de nuevo, las cuatro cargas en la primera iteración consumieron todo el bloque de caché. Cada carga trae un valor de 64 bits (8 bytes). Por lo tanto, el tamaño del bloque es de 32 bytes.

□