

- d) **Calcula** la tasa de fallos en cache que se obtiene al ejecutar completamente el bucle más interno 1 vez y da el resultado en porcentaje.

- e) **Calcula** el total de páginas de Memoria Virtual que se utilizan para ejecutar completamente el bucle más interno 1 vez. Da el resultado para un valor de $M = 512$.

Con el objetivo de reducir los fallos de Carga, a la cache de datos anterior se le añade un mecanismo de prefetch para traer a la cache información antes de que sea solicitada en un futuro cercano. Considerar dos tipos de prefetch: (a) se lanza un prefetch del bloque $i+1$ cada vez que se accede al bloque i (prefetch OBL - one block lookahead), y (b) al observar una secuencia de accesos a bloque b , $b+1*n$, entonces se hace prefetch de $b+2*n$ (prefetch con stride).

- f) **Calcula** la cantidad de fallos totales en cache que se producen al ejecutar completamente el bucle más interno 1 vez en presencia del prefetch OBL. Debes dar el resultado en función de la constante M .

- g) **Calcula** la cantidad de fallos totales en cache que se producen al ejecutar completamente el bucle más interno 1 vez en presencia del prefetch con stride. Debes dar el resultado en función de la constante M .

Suponer que se ejecutan completos los tres bucles del código IJK en un procesador con una cache de tamaño limitado y para un valor de $M = 256$. Al pasarle al ejecutable la herramienta valgrind observamos que se ejecutan 691×10^6 instrucciones dinámicas y al instrumentar el código con GetTime () obtenemos un tiempo de ejecución de 57.85 ms. Además, el código IJK se modifica implementando una ordenación distinta de sus bucles y añadiendo una optimización adicional como la que vimos en la práctica 8. El nuevo código se compila con la opción -O3 y se obtiene un ejecutable que se ejecuta 12 veces más rápido.

- h) **Calcula** el rendimiento en MFLOPS del código original y del código optimizado. Da el resultado en GFLOPS.

COGNOMS:

[illegible]

NOM:

[illegible]

Problema 2. (3,4 puntos)

Queremos estudiar el efecto de la búsqueda y decodificación de instrucciones en el rendimiento de un procesador segmentado superescalar con ejecución fuera de orden de la familia x86. Para ello, simulamos la ejecución de un programa de prueba P en un simulador parametrizable. Para simplificar el problema, ignoraremos las interferencias que puedan causar los accesos a datos y nos centraremos exclusivamente en los accesos a instrucciones, por lo que supondremos que en los accesos a datos nunca se produce un fallo al acceder a la cache de datos.

En los procesadores segmentados pueden transcurrir decenas de ciclos hasta que se calcula la dirección destino del salto y el procesador decide si el salto se producirá (*taken*) o por el contrario se seguirá ejecutando en secuencia (*not taken*). La alternativa más simple consiste en ejecutar por defecto instrucciones de forma secuencial, con lo que los saltos *not taken* no incurren en ninguna penalización. Sin embargo, los saltos *taken* incurren en una penalización de 10 ciclos. A este procesador le llamaremos PSeq.

Con el simulador hemos obtenido que el 20% de las instrucciones dinámicas de P son saltos, y que el 75% de los saltos son *taken*. También hemos obtenido que, en un procesador ideal donde los saltos *taken* no penalizan, el CPI es de 0,5 ciclos/instrucción.

a) **Calcula** el CPI del procesador PSeq.

Ante la pérdida de rendimiento debida a los saltos, se ha decidido incorporar un predictor de saltos al procesador PSeq, con el que hemos obtenido un CPI de 0,9 ciclos/instrucción. En el diseño propuesto, la penalización debida a un salto mal predicho es de 20 ciclos. Los saltos bien predichos no tienen penalización.

b) **Calcula** la tasa de fallos del predictor de saltos.

Finalmente, se ha optado por usar un predictor de saltos más sofisticado que usaremos en los siguientes apartados, que tiene una tasa de aciertos del 95% con el que se obtiene un CPI de 0,7 ciclos/instrucción. Hasta ahora no hemos considerado los fallos en la cache de instrucciones. Los procesadores superescalares leen varias instrucciones cada vez que acceden a la cache de instrucciones, por lo que el número de accesos a la cache es menor que el numero de instrucciones ejecutadas. Sabemos que en P se ejecutan 10×10^9 instrucciones dinámicas y que se realizan 4×10^9 accesos a la cache de instrucciones, es decir, se realizan sólo 0,4 accesos a la cache por instrucción. Sabemos además que la cache de instrucciones tiene una tasa de fallos de 0,05 fallos/acceso y la penalización es de 100 ciclos/fallo.

c) **Calcula** los ciclos perdidos debidos a los fallos en la cache de instrucciones y el CPI real del procesador.

--

El conjunto procesador-cache está conectado a un sistema de memoria principal¹ mediante un único canal de 64 bits al que se ha conectado un DIMM de 4 Gbytes. Este DIMM tiene 8 chips de memoria DDR-SDRAM (Double Data Rate Synchronous DRAM) de un byte de ancho cada uno. El DIMM esta configurado para leer/escribir ráfagas de 64 bytes. La

1.La CPU funciona a una frecuencia interna mayor que la memoria por lo que un ciclo de memoria corresponde a múltiples ciclos de CPU.

latencia de fila es de 5 ciclos, la latencia de columna es de 4 ciclos y la latencia de precarga es de 2 ciclos.

- d) Suponiendo que el comando ACTIVE se lanza en el ciclo 1, **indica** en qué ciclo se lanzará el comando PRECHARGE en una operación de lectura de un bloque de 64 bytes de la memoria DDR-SDRAM. Justifica la respuesta.

Esta DDR funciona a una frecuencia de 1 GHz. La potencia consumida por la DDR depende de su actividad. Durante un acceso a un bloque de 64 bytes se consumen 10 W (desde que se envía el comando ACTIVE hasta que se completa el PRECHARGE), mientras que el resto del tiempo consume sólo 1 W. Sabemos que el programa P ha tardado 9 segundos en ejecutarse y que se han realizado $0,2 \times 10^9$ accesos de 64 bytes a la DDR.

- e) **Calcula** la potencia media consumida por la memoria DDR durante la ejecución del programa P.

Dado que el rendimiento obtenido hasta ahora es bastante pobre, se ha sugerido hacer prefetch de instrucciones. Se sugiere hacer prefetch en caso de fallo, de forma que en cada fallo se leerán dos bloques de 64 bytes cada uno: el que ha provocado el fallo y el siguiente. En este caso, el controlador de memoria envía los comandos necesarios a la DDR-SDRAM de forma que ambos accesos sean servidos lo más rápidamente posible. Supongamos que los dos bloques están en la misma página de DRAM y que la página no está abierta.

- f) Suponiendo que se maximiza el ancho de banda útil del bus y que el comando ACTIVE del primer bloque se lanza en el ciclo 1, **indica** en qué ciclo se lanzará el comando PRECHARGE en una operación de lectura de dos bloques consecutivos de 64 bytes de la memoria DDR-SDRAM. Justifica la respuesta.

Nuestro procesador traduce las instrucciones x86 a microoperaciones RISC (uops). Cada acceso a la cache consume 2,5 nJ (sea acierto o fallo) y cada instrucción x86 gasta 1 nJ en ser decodificada y traducida a uops.

- g) **Calcula** la energía consumida por la búsqueda y traducción de las instrucciones x86 durante la ejecución de P.

Se ha decidido introducir una cache de micro-ops donde se guardan las uops que ya han sido traducidas para que en caso de acierto no sea necesario decodificar y traducir de nuevo las instrucciones x86. Con el simulador se ha medido que para ejecutar P se generan 15×10^9 uops dinámicas y se realizan 5×10^9 accesos a la cache de uops. Gracias a esto, el número de accesos a la cache de instrucciones se ha reducido a $0,4 \times 10^9$ accesos y el número de instrucciones x86 decodificadas se ha reducido a 1×10^9 instrucciones. Un acceso a la cache de uops consume 1,6 nJ.

- h) **Calcula** la energía consumida por el sistema con cache de uops durante la ejecución de P.

Para implementar los diversos sistemas RAID disponemos de un único modelo de disco duro de 3TB.

- d) **Calcula** cuántos discos duros de 3TB necesitaríamos para cada uno de los niveles de RAID 10 (mirror doble), 5 y 6 si queremos almacenar al menos 18TB útiles

Después de explicar las diferencias entre los tres niveles de RAID, la empresa se decanta por montar un RAID5 con 10 discos duros de 3TB. Cada disco duro tiene un MTTF de 100.000 horas, y se calcula que, en caso de fallo, el tiempo necesario para reemplazar un disco físico y reconstruir los datos es de 4 horas.

- e) **Calcula** el MTTF del sistema de almacenamiento para nuestro RAID 5 con 10 discos físicos.

Por último, la empresa nos pide que calculemos el Speedup de las dos fases de acceso a disco con RAID 5 respecto al original, teniendo en cuenta que en la Fase 1 se leen 7.2 TB y en la Fase 3 se escriben 10.8 TB. Se considera que todos los accesos a disco son secuenciales.

- f) **Calcula** el ancho de banda de lectura en la Fase 1 en nuestro sistema RAID 5 con 10 discos físicos.

- g) **Calcula** el ancho de banda de escritura en la Fase 3 en nuestro sistema RAID 5 con 10 discos físicos.

- h) **Calcula** el Speedup y el tiempo total de la aplicación respecto al original (1 core + 1 disco) después de aplicar todas las optimizaciones (RAID 5 con 10 discos + 12 cores + SIMD).