Universidad de Zaragoza

Departamento de Organización de Computadoras

AOC - Proyecto 2

Jerarquía de Memoria

Iván Deza y Víctor Martinez

José L. Briz, Javier Resano y Alejandro Valero

May 5, 2024

Índice

1 Introducción

El proyecto en cuestión trata de el diseño e implementación de una unidad de control que sea capaz de controlar una jerarquía de memoria donde tenemos: una memoria cache, de rápido acceso, una memoria Scratch de datos, también de acceso rápido, además de una memoria principal más lenta que las anteriores mencionadas. Esta jerarquía de memoria viene conectada vía un bus, de tipo semisincrono, que actúa de árbitro entre todas las memorias de la jerarquía. Además la política de funcionamiento de la cache es **Fetch on Write Miss** (traemos el bloque que necesitamos cuando hay un fallo de escritura) y **Copy Back** (no actualizamos escrituras a la memoria principal instantáneamente si no cuando hay un fallo y nos deshacemos de ese bloque). Además de todo esto, la política de reemplazo es de tipo FIFO (First In First Out).

Esta jerarquía de memoria se introduce en el chip MIPS del proyecto número 1, al cual se le ha cambiado el MD subsystem por uno más complejo, el cual incluye una jerarquía de memoria.

Además de esto, en el apartado 8.1, hemos comentado como funciona un ataque de canal lateral a una memoria cache, demostrado su funcionamiento y creado una prueba de concepto para este tipo de ataques. También se propone una estrategia para mitigar esta vulnerabilidad.

2 Diagrama de Estados de la UC

Aquí se adjunta una imagen con el diagrama de estados de la unidad de control que hemos implementado. Este diseño tiene política de escritura **Copy Back** la cual consiste en que no se escribe en memoria principal todos los cambios que se han ido haciendo durante el programa, si no que se actualiza solo en el momento en el que se bloque se expulsa de la memoria cache. Este hace que la carga de trabajo de entrada salida sea menor ya que reducimos en gran medid los accesos de escritura a memoria principal. Además, ya que la memoria cache es más rápida que la memoria principal, hace que se reduzca el tiempo de ejecución de ciertas secuencias de instrucciones como bucles.

Otra politica utilizada es la denominada **Fetch on Write Miss**, la cual consiste en que traemos a cache los datos en caso de que halla un fallo de escritura. Esto complementa muy bien la política copy back y hace que el rendimiento de la cache sea superior a otro conjuntos de politicas como Write Through junto a una política que implemente Fetch on Write miss.Rivoire (1999)

3 Descomposición de la Dirección de la cache

Para direccionar nuestra memoria cache hemos optado por una descomposición en 8 bits, con los cuales podemos direccionar de la siguiente manera: 1 bit para direccionar entre los 2 sets (asociatividad 2), otro bit para direccionar entre las 2 vías que existen en cada set , 2 bits que nos permiten direccionar bloques dentro de las vías, otros 2 bits para direccionar palabras dentro de estos bloques y por ultimo 2 bits para direccionar bytes dentro de la palabra. En nuestro caso como necesitamos direcciones alineadas estos 2 ultimos bits tendran siempre el mismo valor (00).

Diagrama descriptivo del direccionamiento de la cache:

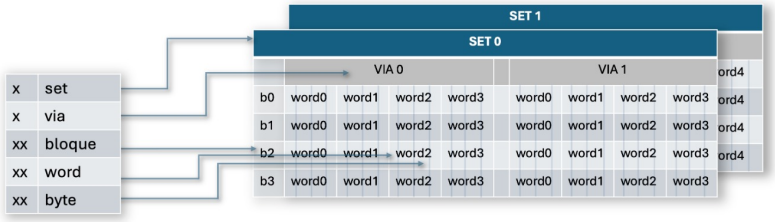

Figure 1: Descomposicion del direccionamiento de la MC

4 Análisis de Latencias

Para medir el rendimiento de la cache hemos utilizado una medición en ciclos, ya que es una medida de tiempo constante. Podríamos comparar la ejecucion del mismo programa en un procesador que no implementa cache y con uno que si lo hace. También podriamos medir los ciclos de reloj que espera la cpu a tener el dato listo. Factores a tener en cuenta sería el numero de fallos tanto en lectura como escritura ya que son estos los que hacen que nuestro procesador tenga que esperar al dato o instrucción, ralentizando la ejecución.Hennessy and Patterson (1993)

Una manera sencilla de calcularlo sería

$$\text{Tiempo de CPU} = (\text{Ciclos de reloj de ejecucion-CPU} + \text{Ciclos de reloj de detención memoria}) \cdot \text{Duración de ciclo de reloj}$$

Para desgeneralizar el asunto podemos separar esta formula en varias para calcular partes individuales necesarias como serían los ciclos de detencion de memoria:

$$\text{Ciclos de reloj de detencion de memoria} = \frac{\text{Acceso a Memoria}}{\text{Programa}} \cdot \text{Frecuencia de fallos} \cdot \text{Penalización de fallos}$$

Finalmente llegamos a la conclusión de que:

$$\text{Tiempo de CPU} = \text{IC} \cdot (\ \text{CPI}_{ejecucion} + \frac{\text{Accesos a Memoria}}{\text{Instrucción}} \cdot \text{Frecuencia de fallos} \cdot \text{Penalización de fallos}) \cdot \text{Tiempo de ciclo de reloj}$$

4.1 Ciclos Efectivos

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

5 Programas de Prueba

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

6 Ejemplo de Código

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus

sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

Etiam ac leo a risus tristique nonummy. Donec dignissim tincidunt nulla. Vestibulum rhoncus molestie odio. Sed lobortis, justo et pretium lobortis, mauris turpis condimentum augue, nec ultricies nibh arcu pretium enim. Nunc purus neque, placerat id, imperdiet sed, pellentesque nec, nisl. Vestibulum imperdiet neque non sem accumsan laoreet. In hac habitasse platea dictumst. Etiam condimentum facilisis libero. Suspendisse in elit quis nisl aliquam dapibus. Pellentesque auctor sapien. Sed egestas sapien nec lectus. Pellentesque vel dui vel neque bibendum viverra. Aliquam porttitor nisl nec pede. Proin mattis libero vel turpis. Donec rutrum mauris et libero. Proin euismod porta felis. Nam lobortis, metus quis elementum commodo, nunc lectus elementum mauris, eget vulputate ligula tellus eu neque. Vivamus eu dolor.

7 Apartados Opcionales

7.1 Hacking Ético

En este apartado hemos querido enseñar como podíamos realizar un ataque DOS (denial of service o denegación de servicio), si sabemos como esta estructurada la memoria cache de un procesador.

Para detectarlo, podríamos usar un contador de Write Missess, y que si ese valor es igual o muy cercano al numero de writes que se esta haciendo, saltar una señal alertando que posiblemente este tipo de ataque esta tendiendo lugar. Para solucionar esto lo que podríamos hacer es implementar un bit de lock, de manera que ciertas instrucciones que podamos marcar, no abandonarían nunca la cache. Mejorando el rendimiento del procesador frente a este tipo de ataques. Otra solución podría ser mandar un halt de manera que terminaríamos este proceso (en caso de que la procesador sea uniciclo). Información sobre este tipo de ataque: Whang (2024).

8 Horas Dedicadas

Horas invertidas en depurar el proyecto en busca de fallos: Iván D.: 0, Víctor M.P: 0
Horas invertidas en entender el enunciado proporcionado: Iván D.: 2, Víctor M.P: 0
Horas invertidas en el desarrollo de la memoria del proyecto: Iván D.: 2, Víctor M.P: 0
Horas invertidas en el diseño de VHDL: Iván D.: 1, Víctor M.P: 0

9 Autoevaluación

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

Nulla mattis luctus nulla. Duis commodo velit at leo. Aliquam vulputate magna et leo. Nam vestibulum ullamcorper leo. Vestibulum condimentum rutrum mauris. Donec id mauris. Morbi molestie justo et pede. Vivamus eget turpis sed nisl cursus tempor. Curabitur mollis sapien condimentum nunc. In wisi nisl, malesuada at, dignissim sit amet, lobortis in, odio. Aenean consequat arcu a ante. Pellentesque porta elit sit amet orci. Etiam at turpis nec elit ultricies imperdiet. Nulla facilisi. In hac habitasse platea dictumst. Suspendisse viverra aliquam risus. Nullam pede justo, molestie nonummy, scelerisque eu, facilisis vel, arcu.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

10 Anexo

10.1 Imágenes

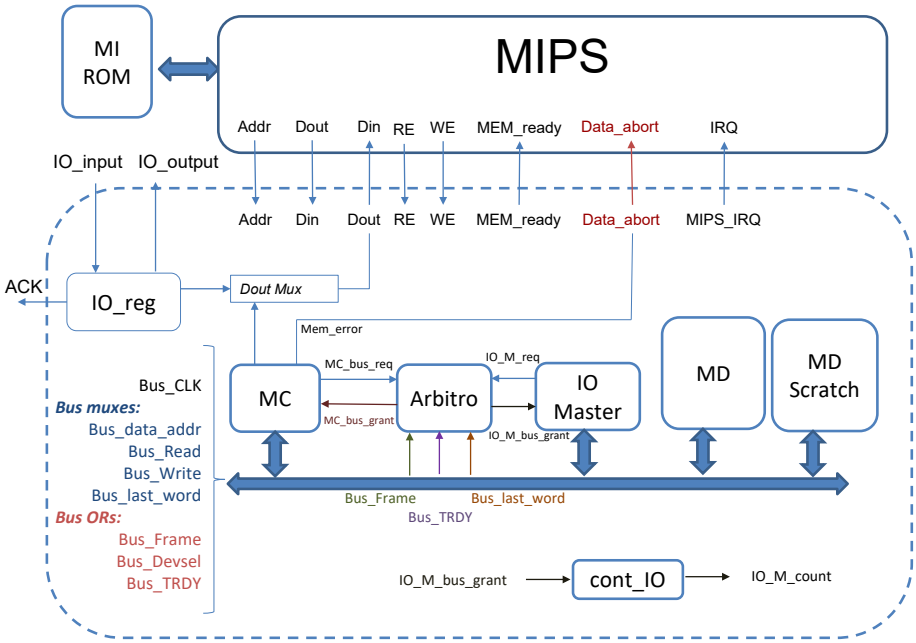

Figure 2: Esquema y señales principales del nuevo IO MD subsytem y de su bus. Se pide diseñar la Unidad de Control de la MC

References

Hennessy, J. L. and Patterson, D. A. (1993). *Arquitectura de Computadores un enfoque cuantitativo*. McGrawHill.

Rivoire, S. (1999). Cache write policies. Consultado el 5 de mayo de 2024.

Whang, Q. (2024). Backcache: Mitigating contention-based cache timing attacks by hiding cache line evictions. pages 1–15, Wuhan, 430072, China. Wuhan University.