SJM: Un simulador de jerarquías de memoria orientado a la docencia de arquitectura de computadores

Francisco J. Alfaro, Aurelio Bermúdez, Pedro J. García, José L. Sánchez
Departamento de Sistemas Informáticos. Universidad de Castilla-La Mancha
Escuela Superior de Ingeniería Informática, 02071, Albacete
{Fco.Alfaro,Aurelio.Bermudez,PedroJavier.Garcia,Jose.SGarcia}@uclm.es

Resumen

El concepto de jerarquía de memoria de un computador es uno de los más importantes en el ámbito de la docencia en arquitectura de computadores. Habitualmente, los distintos aspectos de la estructura y el funcionamiento de la jerarquía de memoria (memoria cache, memoria virtual, algoritmos de reemplazo, tasas de fallos, etc.) se explican primero a los alumnos de manera teórica, y posteriormente se realizan prácticas basadas en programas que simulan dicha jerarquía.

En este trabajo se presenta un nuevo simulador pedagógico, SJM, que permite modelar de forma amigable distintas jerarquías de memoria, modificar fácilmente sus características y obtener medidas de sus prestaciones, de cara a que los alumnos, mediante la evaluación de distintas configuraciones de la jerarquía de memoria, afiancen su comprensión de los distintos aspectos de ésta.

1. Introducción

El concepto de jerarquía de memoria [11] surge en los años 80 para tratar de salvar la cada vez mayor diferencia de velocidad entre los procesadores y las memorias. Una jerarquía de memoria tiene uno o varios niveles de memoria cache (más rápida pero mucho más cara que la habitual memoria principal) que permiten retener los datos más habitualmente utilizados por el procesador para que cuando éste los necesite no sea necesario ir a buscarlos a la lenta memoria principal. Conforme crece la diferencia de velocidad entre los procesadores y la memoria principal, cada vez es más importante sacarle el máximo rendimiento a la jerarquía de memoria del computador.

Así pues, entender bien y saber manejar las distintas configuraciones de una jerarquía de memoria de un computador es cada vez más importante en el ámbito de la docencia en arquitectura de computadores. Habitualmente, los distintos aspectos de la estructura y el funcionamiento de la jerarquía de memoria (memoria cache, memoria virtual, algoritmos de reemplazo, tasas de fallos, etc.) se explican primero a los alumnos de manera teórica, y posteriormente se realizan prácticas basadas en programas que simulan dicha jerarquía.

Como se verá más adelante, desde nuestro punto de vista, ninguno de los simuladores de jerarquía de memoria disponibles actualmente resulta completamente adecuado para ser utilizado en labores docentes. Uno de los principales inconvenientes es que muy pocos de ellos permiten simular configuraciones que integren memoria cache y memoria virtual en una misma jerarquía. Además, muchos de ellos no resultan ni sencillos de utilizar ni amigables, y por tanto dificultan la plena comprensión de las configuraciones modeladas o el análisis de los resultados obtenidos.

En este trabajo se presenta un nuevo simulador pedagógico, que hemos llamado *Simulador de Jerarquías de Memoria* (SJM). Este simulador permite modelar de forma intuitiva distintas jerarquías de memoria, modificar fácilmente sus características y obtener medidas de sus prestaciones, de cara a que los alumnos, mediante la evaluación de distintas configuraciones de la jerarquía de memoria, afiancen su comprensión de los distintos aspectos de ésta. Además, es importante destacar que SJM permite integrar memoria cache y memoria virtual en una misma jerarquía. En cuanto a su aplicación docente, la experiencia de uso nos ha demostrado que el alumno encuentra SJM amigable, útil y sencillo de utilizar.

El resto del artículo está estructurado como sigue: en la sección 2 se ofrece un breve repaso de los principales simuladores disponibles que modelan sistemas jerárquicos de memoria. En la sección 3 se presenta SJM, detallando el procedimiento para configurar una jerarquía, la realización de una simulación a partir de una traza de accesos a memoria y la re-

presentación de los resultados obtenidos. La sección 4 aborda la aplicación de SJM en prácticas y la experiencia de uso. Por último, la sección 5 presenta algunas conclusiones.

2. Simuladores de jerarquías de memoria

Puesto que el sistema de memoria es un componente esencial de un computador, recibe una continua atención tanto por investigadores, para mejorar sus características y nivel de prestaciones, como por docentes, para dar a conocer sus fundamentos y funcionamiento. Es por ello que existen múltiples herramientas desarrolladas por unos y otros con las que pueden lograr sus objetivos más fácilmente. La mayoría de ellas son simuladores de todo o parte del sistema de memoria, y aunque en algunos casos pueden ser usadas tanto para tareas docentes como de investigación, lo habitual es que cada una tenga una determinada orientación. Es normal, por tanto, que su alcance y diseño sea diferente, de tal forma que, por ejemplo, en los simuladores docentes tanto las configuraciones permitidas como los resultados que se manejan tengan ciertas limitaciones, mientras que los orientados a investigación son más sofisticados y admiten mayor diversidad de parámetros de entrada y salida. Por contra, éstos últimos no suelen disponer de interfaces gráficas de usuario, las cuales sí existen en los simuladores docentes, siendo éstas muy cómodas y fáciles de usar, a la vez que ofrecen suficiente información como para poder observar la evolución de las simulaciones con gran detalle y como para entender, a través de diversas imágenes, el funcionamiento del sistema.

Algunos de los simuladores que se han venido usando para estudios en tareas de investigación son, por ejemplo, ACS (*Acme Cache Simulator*), mlcache [13], DRAMsim [16] o Dinero [5]. Este último es un caso especial pues también se ha usado tradicionalmente con propósitos docentes. Es más, este simulador ha sido la base de otros muchos simuladores de este tipo, por lo que merece la pena detenerse para describir sus principales características. Desarrollado por Mark D. Hill, Dinero es un programa de línea de comando que permite analizar detalladamente las prestaciones de un amplio abanico de configuraciones de cache, y requiere una traza con la secuencia de accesos a memoria a simular, la cual puede ser proporcionada en varios formatos. La versión más reciente y completa de este software es DineroIV. Esta versión puede simular varios niveles de cache, distinguir entre caches separadas y unificadas, configurar distintos niveles de asociatividad, tamaños de bloque y de cache, establecer las políticas típicas de escritura en caso de acierto y fallo, fijar las estrategias de asignación y reemplazo más habituales y realizar prebúsquedas. Los resultados que ofrece son los típicos [11]: tasa de fallos, proporción de cada tipo de fallos, tráfico desde y hacia la memoria, etc.

Por otra parte, ejemplos de simuladores principalmente orientados a docencia son Simula Cache (basado en Dinero III) [1], SICAM (simulador de memoria cache multinivel) [2], SIJEM (simulador de jerarquía de memoria) [4], VCS (*Visual Cache Simulator*) [7], MSCSim [9], DCMSim [3], VMS (*Virtual Memory Simulator*) [14] o Xcache (básicamente una interfaz gráfica para Dinero III).

La gran mayoría de herramientas de este tipo se plantean para computadores monoprocesador, pues éstos constituyen el entorno en el que se plantea por primera vez el concepto de memoria en cualquier tipo de estudios de informática. Se pueden encontrar también simuladores del sistema de memoria más enfocados a sistemas multiprocesador. En el caso de SMPs, por ejemplo, se encuentra SMPCache [15], que al margen de las características propias de las caches, también incluye varios protocolos de coherencia *snoopy*, necesarios en ese tipo de computadores. Más sofisticado, y en el lado de la investigación, se puede también citar a GEMS (*General Execution-driven Multiprocessor Simulator*) [8], que está formado por un conjunto de módulos para SIMICS, y que hace posible una simulación más detallada de la jerarquía de memoria en sistemas multiprocesador como SMPs o CMPs.

La mayoría de los simuladores antes mencionados se basan en el uso de trazas que registran los accesos a memoria extraídos de la ejecución de aplicaciones en herramientas o sistemas al margen del simulador. También hay simuladores que permiten ejecutar aplicaciones y obtener directamente a partir de ellas los accesos a memoria que alimentan al modelo de sistema de memoria simulado. Es el caso, por ejemplo, del antes mencionado mlcache, o de las herramientas SpimCache [12] y SpimVista [10], especialmente interesantes. Se trata en ambos casos de extensiones gráficas para el conocido simu-

Parámetro	Rango de valores
Tamaño de cache	mínimo 1, máximo 2^{30} bytes
Tamaño de bloque	mínimo 1, máximo 2^{30} bytes
Niveles de cache	mínimo 1, máximo 4
Bloques por conjunto para cache	mínimo 1 (correspondencia directa), máximo ilimitado, pero los tamaños parciales están limitados por el tamaño de cache y de bloque
Nº de conjuntos para cache	mínimo 1 (correspondencia asociativa), máximo ilimitado, pero los tamaños parciales están limitados por el tamaño de cache y de bloque
Algoritmos de reemplazo para cache	LRU, FIFO y aleatorio
Escritura en caso de acierto en cache	Postescritura y escritura inmediata
Escritura en caso de fallo en cache	Carga en escritura y no carga en escritura
Tipo de acceso a cache	Física, virtual y virtualmente indexada pero físicamente etiquetada
Tipo de cache	Unificada o separada
TLB	Unificado o separado
Número de entradas en TLB	mínimo 1, máximo 50000
Número de entradas de la tabla de páginas	mínimo 1, máximo 5×10^7
Algoritmo de reemplazo de páginas en MV	LRU
Escritura en caso de acierto en MV	Postescritura
Escritura en caso de fallo en MV	Carga en escritura

Tabla 1: Parámetros de una jerarquía de memoria que permite manipular el simulador.

lador Spim [6], desarrollado por James R. Larus, que permiten visualizar la evolución de una determinada configuración de cache al tiempo que se ejecuta un programa escrito en ensamblador de MIPS.

Como se ha indicado, la gran mayoría de los simuladores de la jerarquía de memoria disponibles se centran exclusivamente en los niveles de cache. Unos pocos se ocupan de la memoria principal o la memoria virtual de forma aislada, y muy escasos son los que tratan con el sistema de memoria completo. Según la información que se tiene de estos últimos, ninguno considera caches virtuales o virtualmente indexadas y físicamente etiquetadas. Tampoco, a excepción de los más sofisticados dedicados a investigación, suelen considerar los retardos de los diferentes niveles de la jerarquía.

El simulador que se presenta en este trabajo pretende incorporar todas estas características permitiendo un manejo sencillo e intuitivo al alumno a la vez que proporcionando suficiente nivel de detalle en el modelado de la jerarquía, así como ayuda contextual para solventar las dudas más frecuentes que pudieran surgir. Para ello, se ha partido del simulador DineroIV, al que se ha añadido una interfaz gráfica, soporte para caches virtuales y caches virtualmente indexadas pero físicamente etiquetadas, cálculo del tiempo de acceso, etc.

3. Descripción de SJM

En esta sección se presenta el funcionamiento del simulador que hemos desarrollado. Comenzaremos describiendo la fase de configuración de la jerarquía de memoria para pasar luego a detallar el proceso de simulación de la jerarquía configurada. Por último, se describirá la representación de los resultados obtenidos.

3.1. Configuración de una jerarquía

Las características de las jerarquías de memoria que permite configurar el simulador están recogidas en la tabla 1. En la ventana de configuración (ver figura 1) se pueden ir añadiendo elementos a la jerarquía de memoria, empleando los botones que hay a la derecha. De esta forma, se pueden añadir hasta 4 niveles de memoria cache, un TLB (*Translation Lookaside Buffer*), que podrá ser de datos o instruc-

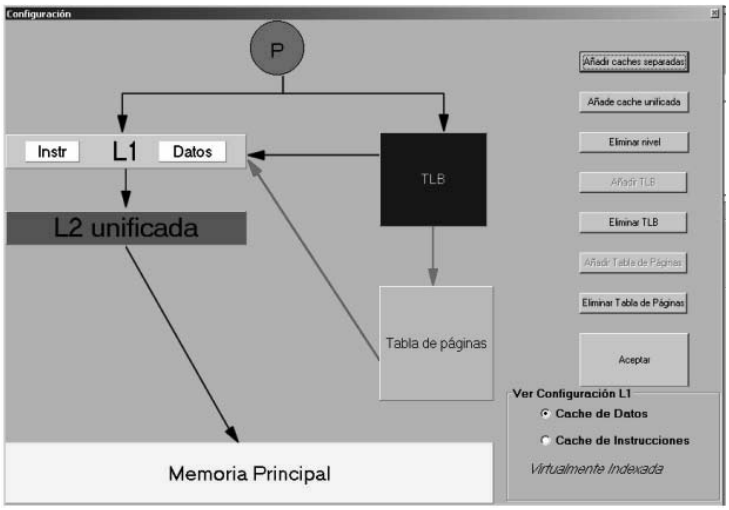

Figura 1: Ventana de configuración de la jerarquía.

ciones, y una tabla de páginas. La inclusión de ciertos elementos hace obligatoria la inserción de otros (por ejemplo un TLB necesita una tabla de páginas) aunque hay ciertas combinaciones que son opcionales. La representación gráfica de la jerarquía en esta ventana dependerá de la configuración escogida.

Seleccionando cualquier elemento de la ventana (excepto en el procesador) se puede acceder a una nueva ventana contextual que permite configurar los parámetros propios del elemento seleccionado. Por ejemplo, en la figura 2 se puede ver la ventana de configuración de la cache de datos de primer nivel. Dentro de cualquiera de las ventanas de cache se pueden configurar todos los parámetros típicos de una cache: asociatividad, algoritmos de reemplazo, políticas de búsqueda, políticas de escritura, tiempo de acceso y tipo de cache. La opción de configuración de tipo de cache (separadas o unificada) sólo aparece en la configuración de cache de nivel 1, pues a partir del segundo nivel de cache siempre tiene que ser unificada.

El simulador necesita los tiempos de acceso de todos los componentes de la jerarquía, así como el tamaño de los distintos elementos. En el caso del TLB es necesario, además, determinar si se accede de forma separada para datos e instrucciones, o bien si consideramos un único TLB tanto para datos como para instrucciones. En la configuración de la tabla de páginas es necesario determinar el número de marcos físicos que se pueden alojar en memoria.

Una vez completa la configuración de la jerarquía, se pueden almacenar todos los datos en un archivo para que pueda ser recuperada posteriormente, agilizándose así la tarea de repetir la simulación en las mismas condiciones.

3.2. Simulación de una traza de accesos

Una vez configurada la jerarquía de memoria a simular, la herramienta muestra una ventana con diversos paneles, que se corresponden con los elementos configurados (figura 3). En esta vista, no es posible modificar las características de ninguno de los elementos definidos. Para cambiar el valor de estos parámetros, sería necesario emplear de nuevo la ventana de configuración previamente descrita.

Los distintos paneles permiten realizar un seguimiento de la evolución de los distintos elementos de la jerarquía a medida que se lleva a cabo la secuencia de accesos a memoria durante la ejecución de la simulación. La dirección virtual correspondiente a cada acceso se muestra en la parte superior de la ventana principal. En la parte inferior de la ventana de simulación se muestran unos paneles que represen-

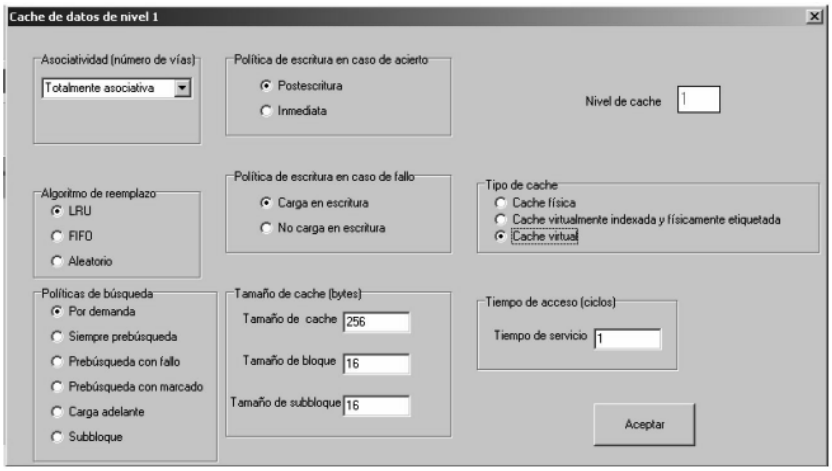

Figura 2: Configuración de la cache.

tan las caches definidas, organizadas en conjuntos y en bloques. Según el tipo escogido, se mostrarán caches de instrucciones, de datos o unificadas.

Seleccionando cualquiera de las caches aparecerá una ventana flotante que muestra de forma más detallada el formato de dirección que utilizará la cache (indicando cuáles son los campos etiqueta, índice y desplazamiento) y su estado en cada momento. En concreto, los bloques que han sido sustituidos se resaltan empleando un código de colores. Todas estas ventanas pueden manipularse y organizarse en cascada, de forma horizontal, vertical o a gusto del usuario. En la configuración mostrada en la figura 3, se observa que para el primer nivel la cache de instrucciones es de correspondencia directa (sólo tiene un bloque por línea), mientras que la cache de datos es asociativa de grado dos (dos bloques por línea de cache). Para el segundo nivel, puede observarse como la cache es unificada y asociativa de grado cuatro (4 bloques por línea de cache).

Según el tipo de configuración se mostrará una cantidad distinta de datos. Así, por ejemplo, en el caso de una cache de prebúsqueda se representarán todos los bloques traídos de cache utilizando una flecha doble y un color distinto para mostrar la diferencia entre los dos tipos de acceso.

Por su parte, el TLB y la tabla de páginas se muestran en forma de tabla en la zona central de la ventana de simulación. En dichas tablas, además de la correspondencia entre página virtual y página física, se indica el valor de los bits de validez y suciedad y el valor del contador mediante el que se implementa el algoritmo de reemplazo LRU. Bajo el TLB y la tabla de páginas se muestra la dirección física correspondiente a la dirección virtual actual. Esta vista también recoge en todo momento los ciclos de simulación consumidos.

Para facilitar el manejo del simulador, todos los elementos de la ventana tienen acceso a una ventana contextual de ayuda, de forma que el usuario pueda resolver rápidamente sus dudas. Además, en la parte superior de la aplicación se incluye una barra de herramientas que permite acceder de forma rápida a todas las opciones de los menús.

Para realizar la simulación de una traza de accesos a memoria sobre la jerarquía definida, una vez cargado el fichero de traza, ésta se muestra en la parte superior izquierda de la ventana de simulación (ver figura 3). En concreto, una tabla nos informa sobre el tipo de acceso (instrucción, lectura de datos o escritura) y la dirección correspondiente a cada acceso. Durante la simulación esta tabla informa sobre el próximo acceso que se va a realizar.

La simulación se puede llevar a cabo de dos formas: paso a paso o completamente. La simulación paso a paso permite ir estudiando las repercusiones

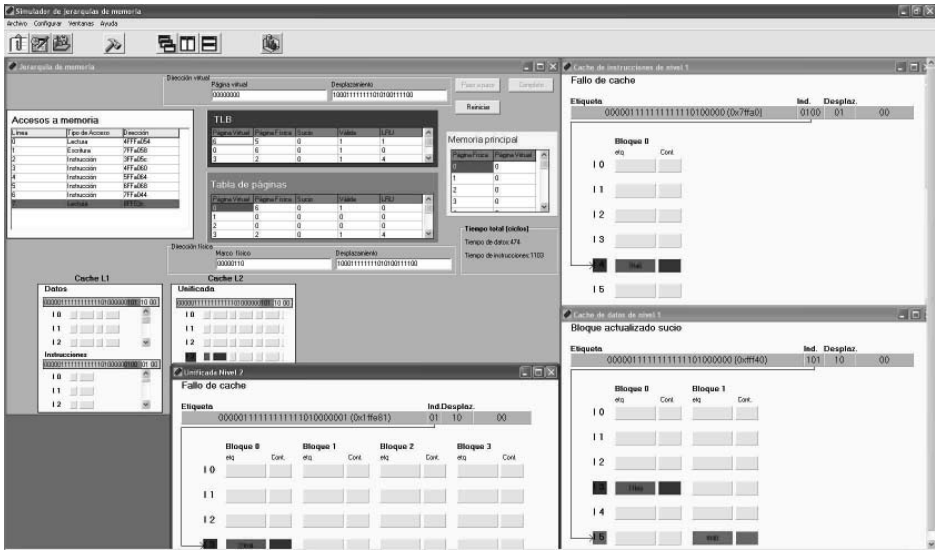

Figura 3: Ventanas de la simulación.

de cada nuevo acceso sobre los niveles de la jerarquía de memoria y sobre el tiempo total. Por otra parte, la simulación completa se detiene sólo cuando se han realizado todos los accesos de la traza.

3.3. Representación de los resultados

Independientemente del tipo de simulación empleado (paso a paso o completa), una vez finalizada ésta, la propia ventana de simulación proporciona algunos datos interesantes. En particular, se puede observar (figura 3) el estado final de las caches, la tabla de páginas y el TLB, así como el tiempo total invertido, diferenciando entre accesos a datos y accesos a instrucciones. Además, en la ventana de estadísticas (figura 4) está disponible el conjunto completo de resultados obtenidos por la simulación. En concreto, se proporcionan resultados para cada nivel de la jerarquía de memoria, así como los tiempos de acceso a datos, instrucciones y global. Finalmente, se proporciona la posibilidad de almacenar en archivo o imprimir el informe generado.

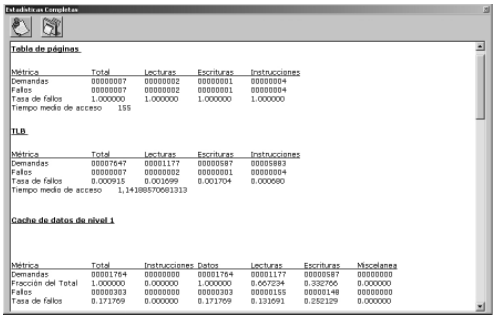

Figura 4: Ventana de resultados.

4. Aplicación docente y experiencia de uso

Como ya hemos mencionado, SJM ha sido desarrollado con el principal objetivo de emplearlo en la docencia de aquellas asignaturas que se ocupan, a distintos niveles, del sistema de memoria de un computador, y más concretamente en las prácticas que deben servir para reforzar y complementar los conceptos teóricos introducidos en dichas asignaturas. Evidentemente, este objetivo básico ha condicionado el diseño del simulador, tal y como se ha explicado en anteriores secciones. Sin embargo, y aunque en nuestra opinión el resultado final del simulador se

ajusta efectivamente a los planteamientos iniciales de su diseño, esto no garantiza directamente el éxito de su aplicación en la docencia. En este sentido, y de cara a obtener un máximo partido de esta herramienta desde el punto de vista docente, convendría tener en cuenta ciertas consideraciones y reflexiones, fruto tanto del análisis "a priori" de las funcionalidades de SJM como de las experiencias de su uso en el aula.

En primer lugar, cabe recordar que SJM permite simular simultáneamente todos los niveles posibles de una jerarquía de memoria (cache, memoria virtual, etc.), así como configuraciones diversas dentro de ciertos niveles (por ejemplo, caches separadas en instrucciones y datos, unificadas, etc.). Sin embargo, los conceptos relativos a los distintos niveles de la jerarquía suelen introducirse gradualmente a los alumnos, de modo que es muy probable que el primer contacto de éstos con SJM se produzca antes de que puedan comprender algunas de las posibilidades de modelado de jerarquía que ofrece la herramienta. Por tanto, en este sentido, conviene tomar la precaución elemental de preparar las prácticas de modo que los alumnos se adentren en las funcionalidades y opciones de la herramienta de forma paralela a su progresión en su capacidad de diseño y análisis de jerarquías de memoria. Concretamente, lo habitual es que los alumnos se aproximen en primer lugar a los niveles superiores de la jerarquía de memoria (básicamente, memoria cache y principal), por lo que las primeras prácticas a desarrollar con SJM deberían plantearse (como es nuestro caso) sin usar más funcionalidades que las estrictamente necesarias para simular estas configuraciones simples. Ahora bien, una vez que los alumnos poseen una visión de todos los posibles niveles y opciones de la jerarquía de memoria, SJM está igualmente preparado para simular cualquier configuración completa de jerarquía.

Independientemente del grado de complejidad de la jerarquía simulada, conviene tener presente que existe un riesgo para el auténtico aprovechamiento de las prácticas realizadas con SJM (y en general, con cualquier simulador similar). Dicho riesgo consiste en que los alumnos, aunque lleguen a manejar con soltura la herramienta durante las prácticas, no extraigan de ellas conclusiones que afiancen sus conocimientos y refuercen su capacidad de diseñar y analizar jerarquías de memoria. Este riesgo debería evitarse planteando las prácticas, como en nuestro caso, desde un punto de vista causa-efecto en la relación diseño-resultado. Esto es, no debe bastar con que el alumno sea capaz de configurar una jerarquía de memoria que se plantea y de obtener los resultados de simular la misma, ni siquiera con que el alumno aprecie diferencias entre los resultados de unas jerarquías respecto a otras, sino que, idealmente, debe llegarse al punto en el que el alumno sea capaz, a partir de los resultados obtenidos, de deducir por qué se producen dichas diferencias en los resultados, teniendo en cuenta las jerarquías modeladas. Como alternativa, las prácticas pueden plantearse desde el punto de vista inverso, de modo que el alumno deba diseñar una jerarquía que obtenga determinados resultados (relativos a otras o absolutos). Igualmente, y teniendo en cuenta que las simulaciones con SJM se basan en trazas de acceso a memoria, resulta interesante plantear que los alumnos creen sus propias trazas de modo que produzcan un determinado resultado para una configuración concreta de jerarquía de memoria. Como puede comprobarse, en ninguno de los planteamientos expuestos es posible que el alumno complete las prácticas con éxito sin demostrar cierto dominio de la jerarquía de memoria.

Por experiencia, hemos comprobado que, para todos los mencionados enfoques de las prácticas, las funcionalidades de SJM resultan completamente adecuadas. En este sentido, los alumnos suelen encontrar especialmente útil y amigable la interfaz de usuario, lo que facilita sin duda el desarrollo de las prácticas, durante las cuales los alumnos pueden centrarse en el diseño de la jerarquía y en el análisis de los resultados sin grandes pérdidas de tiempo debidas a la operación de la herramienta.

Por último, cabe mencionar que el hecho de que los alumnos puedan disponer libremente del simulador (para instalarlo en sus equipos particulares), junto con la sencillez de su uso, les permite practicar con el mismo más allá del tiempo presencial en el laboratorio. En nuestro caso, solemos animar a los alumnos a que diseñen de este modo sus propias configuraciones de jerarquía de memoria y realicen los correspondientes análisis de resultados.

5. Conclusiones

Uno de los aspectos fundamentales de los tratados en la docencia en arquitectura de computadores es la

jerarquía de memoria. Habitualmente, el alumno adquiere las competencias relacionadas con dicho concepto a través de un conjunto de actividades, entre las que son parte fundamental las prácticas basadas, en general, en el uso de uno o varios simuladores que modelan la jerarquía de memoria.

En este artículo se ha hecho un repaso de los principales simuladores disponibles para esta tarea, de los que ninguno, desde nuestro punto de vista, reúne todas las características que serían deseables en una herramienta orientada a la docencia: sencillez, amigabilidad, completitud, integrar memoria cache y memoria virtual, proporcionar tiempos de acceso, etc.

En este artículo hemos descrito el simulador SJM, que permite modelar de forma sencilla una jerarquía de memoria que incluya uno o varios niveles de cache, memoria virtual, tabla de páginas, TLB, etc. También hemos presentado reflexiones sobre su adecuado empleo en prácticas, y sobre la experiencia de uso en las asignaturas en las que venimos utilizando este simulador, que los alumnos encuentran útil a la vez que sencillo y amigable.

Referencias

[1] Almisas, R., Paz, R., Linares, A., Amaya, C., Sevillano, J.L., *Un simulador de memorias cache multinivel*, Actas de las Jornadas de Enseñanza Universitaria de la Informática (JENUI), 2001.

[2] Benavides, J.I., Herruzo, E., Bandera, G., *Aplicación para la Simulación del Comportamiento de una Memoria Cache Multinivel*, Actas de las XIII Jornadas de Paralelismo, 2002.

[3] Cordeiro, E.S., Stefani, I.G.A., Soares, T., Martins, C.A.P., *Dcmsim: Didactic cache memory simulator*, Frontiers in Education Conference (FIE03), 2003.

[4] Departamento de Ingeniería de Sistemas y Automática y Arquitectura de Computadores de la Universidad de La Laguna, *Simulador de Jerarquía de Memoria*, `http://www.isaatc.ull.es/portal/proyectos/sijem/`.

[5] Hill, M.D., *Dinero IV Trace-Driven Uniprocessor Cache Simulator*, `http://pages.cs.wisc.edu/~markhill/DineroIV/`.

[6] Larus, J.R., *SPIM. A MIPS32 Simulator*, `http://pages.cs.wisc.edu/~larus/spim.html`.

[7] Leahy, B., *Visual cache simulator*, `http://www.cc.gatech.edu/fac/Bill.Leahy/vcsa/vcs.html`.

[8] Martin, M.K., Sorin, D.J., Beckmann, B.M., Marty, M.R., Xu, M. Alameldeen, A.R., Moore, K.E., Hill, M.D., Wood, D.A., *Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset*, SIGARCH Comput. Archit. News, 2005.

[9] Mendes, J.L.D., Coutinho, L.M., Martins, C.A.P., *MSCSIM: Multilevel and split cache simulator*. 36th Frontiers in Education Conference (FIE), 2006.

[10] Pascual, L., Torrentí, A., Sahuquillo, J., Flich, J., *Understanding Cache Hierarchy Interactions with a Program-Driven Simulator*, Proceedings of the 2007 Workshop on Computer Architecture Education, 2007.

[11] Patterson, D.A., Hennessy, J.L., *Computer organization and design: the hardware/software interface*. Elsevier/Morgan Kaufmann, fourth edition, 2008.

[12] Petit, S., Tomás, N., Sahuquillo, J., Pont, A., *An Execution-driven Simulation Tool for Teaching Cache Memories in Introductory Computer Organization Courses*, Proceedings of the 2006 Workshop on Computer Architecture Education, 2006.

[13] Tam, E., Rivers, J., Tyson, G., Davidson, E., *mlcache: A flexible multi-lateral cache simulator*, Sixth IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'98), IEEE Computer Society, 1998.

[14] Tran, N., Menasce, D.A., *Virtual memory simulator*, `http://cs.gmu.edu/cne/workbenches/vmsim/vm.html`.

[15] Vega, M.A., Martín, R., Zarallo, F.A., Sánchez, J.M., Gómez, J.A., *SMPcache: Simulador de Sistemas de Memoria Caché en Multiprocesadores Simétricos*, Actas de las XI Jornadas de Paralelismo, 2000.

[16] Wang, D., Ganesh, B., Tuaycharoen, N., Baynes, K., Jaleel, A., Jacob, B., *Dramsim: A memory-system simulator*, SIGARCH Computer Architecture News, 2005.