SIMDE: Un Simulador para el Apoyo Docente en la Enseñanza de las Arquitecturas ILP con Planificación Dinámica y Estática

Castilla I., Moreno L., Sigut J., González C., González E. J.
Departamento de Física Fundamental y Experimental, Electrónica y Sistemas
Universidad de La Laguna
La Laguna 38271, Tenerife
E-mail: {ivan, lorenzo, jfsigut, evelio}@cyc.ull.es

Resumen

En este trabajo se presenta una herramienta de simulación de arquitecturas ILP en sus dos vertientes más significativas: superescalar (con planificación dinámica) y VLIW (con planificación estática), con el objetivo de incorporarse como herramienta docente en una asignatura de Arquitectura de Computadoras.

1. Introducción

La utilización de simuladores en la docencia de asignaturas de Arquitectura de Computadores es una práctica muy habitual debido a la complejidad de conectar la teoría de muchos de los temas que se tratan con la experiencia práctica. Encontramos simuladores en prácticamente todos los temas de esta asignatura: "Dinero"[1] (cachés), "WinMIPS64"[11] (RISC), "WinDLXV"[10] (vectorial), "MIDAS"[5] o "SATSim"[12] (superescalares), "TMS320C6xxx"[8] (VLIW)… Sin embargo, algunos de ellos no funcionan correctamente, caso del MIDAS con el que sólo se pudo contar con una versión beta; y otros, como el TMS320C6xxx son muy específicos.

Por estas razones, se ha abordado la realización de un simulador que incluya las dos arquitecturas ILP (*Instruction Level Parallelism*): superescalar y VLIW (*Very Long Instruction Word*), con el objetivo de que forme parte de una herramienta docente para incorporar al proceso de enseñanza-aprendizaje de ambas arquitecturas. La combinación en un único programa de los dos enfoques ILP permite resaltar sus diferencias pero también una visión más global de estas técnicas.

El simulador está disponible para su evaluación en: http://www.cyc.ull.es/simde.

2. Modelización de las arquitecturas

Se ha hecho especial hincapié en el diseño de una base común a las dos arquitecturas (Superescalar y VLIW) que simplifique su modelado. Partiendo de esa base se han ido añadiendo los componentes específicos de cada arquitectura.

2.1. Elementos comunes

Los componentes que comparten las dos arquitecturas son:

- Memoria. De un tamaño fijo, está diseñada de manera transparente al resto de la máquina. Simplemente permite operaciones de lectura (*LOAD*) que pueden ser inmediatas o producir un fallo (*miss*); y operaciones de escritura (*STORE*).
- Bancos de registros de punto flotante (FPR) y de propósito general (GPR).
- Unidades funcionales (UF): Se emplean 6 tipos: suma entera, multiplicación entera, suma flotante, multiplicación flotante, memoria y salto. Son segmentadas, con etapas de un ciclo de duración. El número de etapas del *pipeline* determina por tanto el tiempo que tarda en ejecutarse una instrucción en esa UF. Se define la *latencia* de una UF como el número mínimo de ciclos que debe esperar una instrucción que tiene una dependencia verdadera con otra instrucción que le precede antes de ponerla a ejecutar. Este tiempo coincide con el número de etapas del *pipeline*.
- Repertorio de instrucciones: Se ha empleado un subconjunto de instrucciones inspirado en el repertorio MIPS IV. En el caso de la máquina VLIW nos referiremos a las instrucciones como *operaciones* para evitar confusiones con la *instrucción larga*.

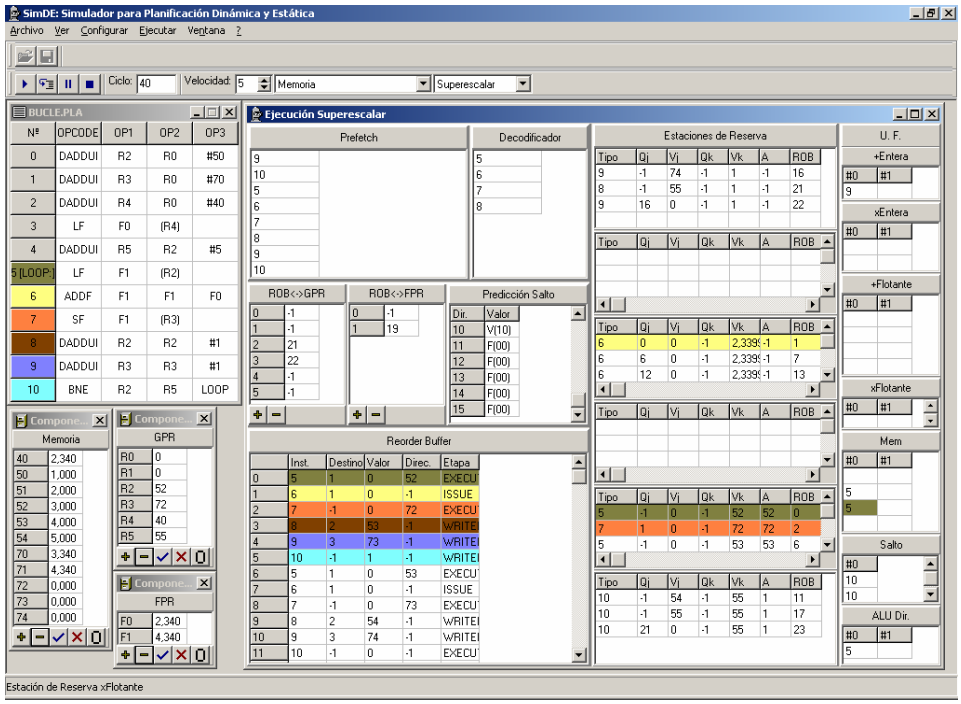

Figura 1. Captura de pantalla de la aplicación efectuando simulación Superescalar. Aparece el código de un programa, los bancos de registros, memoria, ER, UF, ROB, Tabla de Predicción de saltos…

2.2. Máquina Superescalar

La máquina superescalar tiene una serie de funcionalidades añadidas:

- Predicción de salto mediante esquema de 2 bits.
- Se emplea el algoritmo de Tomasulo [2] para el control. La mayoría de los componentes añadidos a la máquina responden a necesidades de este algoritmo.
- *Renaming* integrado en el *Reorder Buffer* [9].

Los componentes que se han añadido al esqueleto básico de la máquina son:

- Unidad de Prebúsqueda (*Prefetch*): Carga las instrucciones del camino predicho del salto de manera transparente al resto del esquema
- *Decoder*
- Estaciones de Reserva (ER). Existe un conjunto de entradas por cada tipo de UF.
- *Reorder Buffer* (ROB).
- Se añaden UF de suma entera adicionales para el cálculo de direcciones de memoria.
- Tabla de predicción de salto de 2 bits

2.3. Máquina VLIW

En el caso de la máquina VLIW se dispone de un hardware muchísimo más simple. Se ha intentado incidir en esta simplicidad para resaltar aún más las diferencias con la máquina Superescalar.

- Palabra de *instrucción larga*: Se ha diseñado totalmente horizontal, con capacidad para tantas operaciones como recursos (UF) haya. La única limitación es una única operación de salto por instrucción.
- No se ha implementado ningún mecanismo de predicción de salto. Sin embargo sí se ha añadido *predicación* [3] asociando a cada operación un *registro de predicado*. De esta manera se pueden planificar operaciones de las dos ramas de un *branch*.
- Se dispone de bits de *NaT* asociados a cada registro de propósito general y punto flotante para indicar que ha habido un fallo de caché al efectuar un *LOAD* que escribía en ese registro.

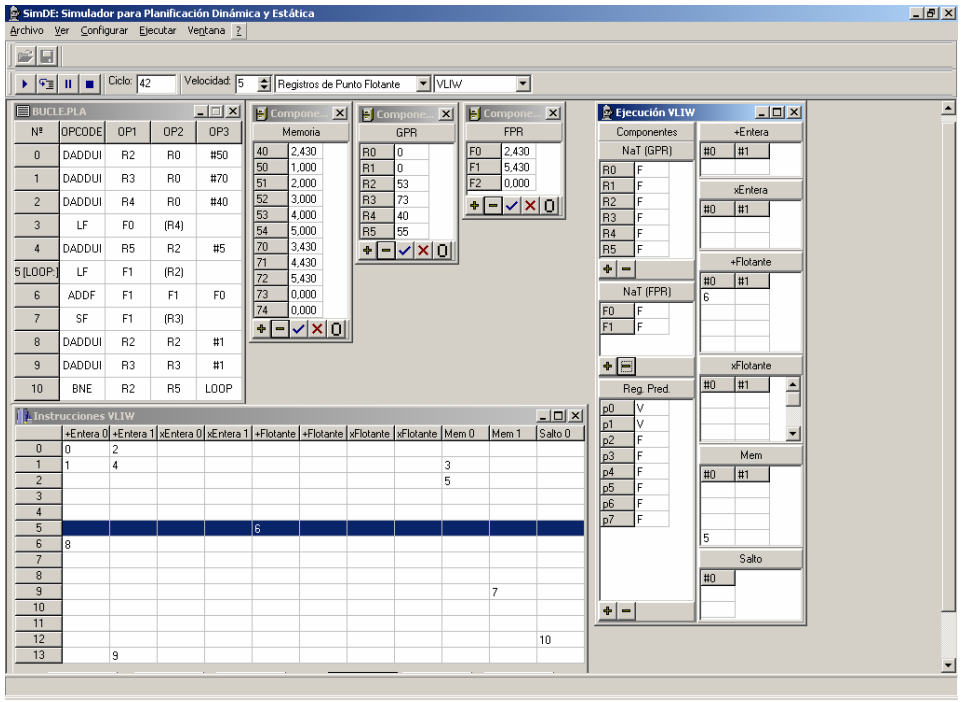

Figura 2. Captura de pantalla de la aplicación efectuando simulación VLIW. Se observan el código secuencial, el código de instrucciones largas (esquina inferior izquierda), bancos de registros, memoria, UF, registros de predicado, marcas NaT…

3. Prestaciones del simulador

Las características más destacables del simulador son:

- Interfaz gráfica compuesta de rejillas que representan el esquema de cada máquina y permiten una visión sencilla del funcionamiento de las mismas (ver pantallas en figuras 1 y 2).
- El programa (figura 3) permite cargar un código secuencial cada vez y realizar simulaciones (superescalar o VLIW) con él.
- El programa secuencial se crea con cualquier editor de texto.
- Construcción interactiva de los programas VLIW partiendo del código secuencial [13].
- Ejecución continua o paso a paso, permitiendo el uso de *breakpoints*.

Son susceptibles de ser cambiados por el usuario los siguientes parámetros de las máquinas:

- Número de UF de cada tipo
- Latencia de cada tipo de UF
- Número de instrucciones emitidas por ciclo en la máquina Superescalar.
- Porcentaje de fallos en la caché de datos.

4. Ampliaciones futuras

El simulador, desarrollado con *C++ Builder*, se ha procurado mantener lo más modularizado posible, de tal manera que cualquier ampliación posterior pueda ser incluida sin tener que rehacer la estructura básica de las máquinas.

Algunas de las ampliaciones contempladas son:

- Implementar técnicas para el procesamiento de múltiples *branches* por ciclo
- Diseñar en detalle una jerarquía de memoria con varios niveles de caché y que permita ilustrar los mecanismos que provocan los fallos de caché.
- *Trace cache* [6] y otros mecanismos que faciliten la emisión múltiple de instrucciones.

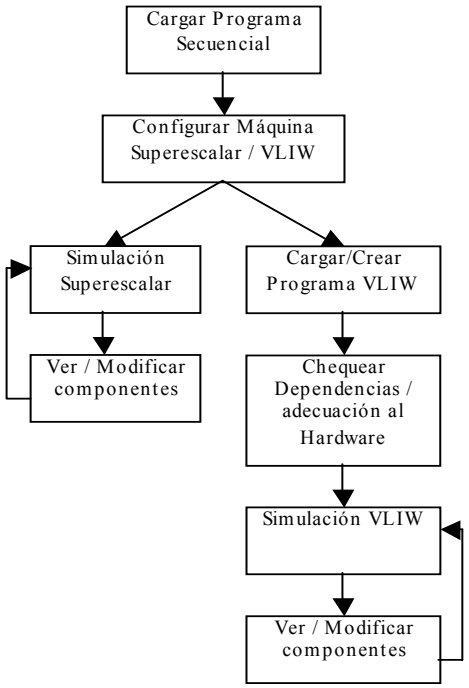

Figura 3. Uso básico del programa

5. Validación

Se puso a disposición de los alumnos de la asignatura de "Arquitectura e Ing. de Computadores" de 5º curso de Ingeniería Informática la herramienta de simulación junto a una encuesta que incluía el perfil del alumno y distintos apartados relativos a los aspectos educativos y técnicos del software. En el momento de la redacción de este artículo aún se están recibiendo los resultados de las encuestas, aunque ya se puede destacar la buena acogida de la herramienta entre los alumnos, que destacan el aspecto pedagógico del simulador.

6. Conclusiones

Se ha presentado un simulador con interfaz gráfica de arquitecturas ILP: superescalar y VLIW, empleando una base común que simplifique el modelado. Para resaltar más la diferencia de planteamiento de ambas concepciones de máquinas, se ha elegido una máquina convencional tipo R10000 [4] y un esquema clásico VLIW tipo Trace 7/200 [7] (aunque incorporando la característica de predicado de la máquina Itanium).

Referencias

[1] *Dinero IV: Trace-Driven Uniprocessor Cache Simulator*, desarrollado en Univ. of Wisconsin Computer Sciences por Edler J., Hill M.. http://www.cs.wisc.edu/~markhill/DineroIV

[2] Hennesy J.L., Patterson D.A., *Computer Architecture: A Quantitative Approach (3rd edition)*, Morgan Kaufmann, 2003

[3] Huck J., Morris D., Ross J., Knies A., Mulder H., Zahir R., *Introducing the IA-64 Architecture*, IEEE MICRO, Sept-Oct 2000

[4] K.C. Yeager, *The MIPS R10000 Superscalar Microprocessor*, IEEE Micro, vol. 16, nº. 2, págs. 28-40, Abril 1996.

[5] *MIDAS Beta-1*, desarrollado en Institut für Technische Informatik, Vienna Technical University por Silhan J., Fuss C. http://icaro.eii.us.es/descargas/R10kSim.zip

[6] Patel S.J., *Trace Cache Design for Wide-Issue Superscalar Processors*, PhD Thesis, The University of Michigan, 1999

[7] Sima D., Fountain T., Kacsuk P., *Advanced Computer Architectures, A Design Space Approach*, Addison-Wesley, 1998

[8] Simulador para la familia de procesadores TMS320C6xxx de Texas Instruments, 2001

[9] Smith, J. E. and Sohi, G. S., *The Microarchitecture of Superscalar Processors*, Proc. of the IEEE, Diciembre 1995

[10] *WinDLXV*, desarrollado en Universidad Politécnica de Valencia por Lopez P., Calpé R.

[11] WinMIPS64, desarrollado por Scott M. http://www.computing.dcu.ie/~mike/winmips64.html

[12] Wolff M., Wills L., *SATSim: A Superscalar Architecture Trace Simulator Using Interactive Animation*, Workshop on Computer Architecture Education, Junio 2000. http://www.ece.gatech.edu/research/pica/SATSim/satsim.html

[13] Zima,H.P., Chapman,B.M., *Supercompilers for Parallel and Vector Computers*, ACM Press Frontier Series/Addison-Wesley, 1990