Simula3MS: simulador pedagógico de un procesador

Raquel Concheiro, Marta Loureiro, Margarita Amor y Patricia González
Dtpo. Electrónica y Sistemas
Universidade da Coruña
15071 A Coruña

Resumen

En este artículo se presenta un nuevo simulador pedagógico, *Simula3MS*, de un procesador para su utilización en asignaturas de arquitectura y tecnología de computadores en cualquiera de las titulaciones de informática. Se presentan los principales simuladores de procesadores utilizados en universidades españolas y se discuten sus ventajas e inconvenientes y el porqué de la necesidad de plantear este nuevo simulador.

1. Introducción

El objetivo de las primeras asignaturas de arquitectura y tecnología de computadores en las titulaciones de informática es introducir y analizar los principios básicos de la organización de los computadores clásicos o von Neumann. Centrándose fundamentalmente en el estudio de los bloques funcionales básicos que componen un sistema monoprocesador: el procesador, la unidad de entrada/salida, la memoria y el sistema de interconexión. Frecuentemente el funcionamiento del procesador es explicado a través de un procesador concreto para facilitar la adquisición de los conceptos, y se utiliza como elemento de apoyo un simulador del procesador.

En este artículo analizamos las características básicas que debe tener un simulador pedagógico de un procesador. Además, se muestran los simuladores utilizados actualmente en docencia, así como sus principales ventajas e inconvenientes, y la necesidad de plantear un nuevo simulador que cubra estas carencias.

El simulador que hemos desarrollado es el *Simula3MS* [13] que simula un procesador MIPS R2000 [12], uno de los procesadores más utilizado como ejemplo en el estudio del procesador. Este nuevo simulador es configurable, con diferentes técnicas segmentadas y no segmentadas, que permite aplicar los conceptos que se imparten en el estudio del procesador. Este trabajo se ha desarrollado en dos proyectos fin de carrera ([6], [11]) de Ingeniería Técnica en Informática de Sistemas. La parte desarrollada en estos proyectos se corresponde al simulador de un procesador con configuraciones monociclo, multiciclo y un procesador segmentado básico.

Este trabajo forma parte de un proyecto global más amplio que comprende la realización de un simulador más completo. Pretendemos ampliar esta herramienta añadiendo, entre otros, la unidad de punto flotante, y métodos de segmentación avanzada, como los algoritmos de Marcador y de Tomasulo.

En la sección 2 se presentan las características básicas que debe tener un simulador pedagógico de un procesador y los principales simuladores utilizados. La sección 3 presenta el nuevo simulador *Simula3MS*. En la sección 4 se describe la experiencia de la utilización del simulador en las prácticas de una asignatura de Tecnología y Arquitectura de Computadores durante el curso académico 2004-2005. Las conclusiones y el trabajo futuro se presentan en la sección 5.

2. Simuladores de procesadores

El uso de simuladores [14, 16] en lugar de procesadores reales resulta bastante común en docencia debido a que ofrecen un entorno de programación más amigable que una máquina real. Además, en los simuladores se pueden detectar más errores, ofrecen más posibilidades que un ordenador real y no modifican elementos físicos del computador.

Por otro lado, los simuladores son una herramienta útil para estudiar la arquitectura de los procesadores y los programas que se ejecutan en ellos. Debido a que están realizados en software y no en silicio,

los simuladores se pueden modificar fácilmente para añadir nuevas instrucciones, construir nuevos sistemas como multiprocesadores o simplemente ofrecer más información.

Finalmente otra razón para usar un simulador en lugar de estaciones de trabajo es que estas no están disponibles universalmente y además el constante progreso hacia máquinas cada vez más rápidas puede dejar estas estaciones obsoletas.

2.1. Características de un simulador

Un simulador pedagógico de un procesador debe permitir observar la evolución de la memoria y de los registros durante la ejecución de las instrucciones. Esto también permite que los alumnos comprueben la traducción de las instrucciones en lenguaje máquina. También es conveniente una representación gráfica del camino de datos con la representación concreta de cada instrucción. Así como la posibilidad de que se pueda ejecutar paso a paso cada instrucción.

Adicionalmente, la inclusión de un editor propio en el simulador facilita a los alumnos la elaboración de los programas ensamblador.

Finalmente, el simulador debería ser configurable con diferentes técnicas segmentadas y no segmentadas para que permita a los alumnos en un único simulador aplicar todos los conocimientos que se les imparten. La presencia de las distintas implementaciones permitirá observar las diferencias de un mismo código según las características del procesador.

2.2. Ejemplos de simuladores pedagógicos

En la actualidad existen varios simuladores de lenguaje ensamblador con características muy diferentes, los más utilizados son el Spim [4, 10] y el DLX [3]. Otros simuladores no tan utilizados son el SimuProc [1] y el Simplescalar [2, 5], más orientado a investigación en arquitectura de computadores.

La principal ventaja del **Spim** [4, 10] radica en estar basado en un procesador real. Este simulador permite observar la evolución de la memoria y de los registros durante la ejecución de las instrucciones. Algunas de las principales carencias de este simulador son la ausencia de una representación gráfica del camino de datos sobre el que se pueda visualizar la ejecución concreta de cada instrucción, o la imposibilidad de observar la ejecución de una instrucción por pasos, o ver los posibles efectos de la segmentación sobre el mismo código. Además no tiene un editor propio, lo que conlleva dificultades para hacer cambios en el código y tener que cargar el fichero después de cada cambio.

Hay otro tipo de simuladores, como el **DLX** [3] o el **SimuProc** [1], que implementan un procesador hipotético, es decir, su repertorio de instrucciones no está basado en ningún procesador real.

El **DLX** es un simulador de un procesador segmentado que permite ver la representación de la ejecución de cada instrucción sobre el camino de datos. Es uno de los más utilizados para docencia. Se pueden configurar las características del procesador, escogiendo entre varias técnicas de segmentación: básico, Tomasulo y Marcador. La configuración, por parte del usuario, de las características del simulador resulta poco amena y nada intuitiva. Uno de sus principales inconveniente es que no incluye una representación de la evolución del segmento de datos o de los registros durante la ejecución del código. En su versión para Windows, **WinDLX** incluye una representación del valor de los registros en cada instrucción.

3. *Simula3MS*: Simulador Pedagógico

Simula3MS es un simulador cuya principal utilidad es su uso pedagógico por lo que el diseño y la implementación de la herramienta ha intentado paliar los defectos detectados en otras herramientas similares. Y aunque *Simula3MS* es ya un simulador útil, aún está en desarrollo.

3.1. Características básicas

Simula3MS es un simulador de un procesador RISC (*Reduced Instruction Set Computer*) [15, 12], que implementa un subconjunto de instrucciones basadas en el repertorio de instrucciones del procesador MIPS [12]. La elección de un procesador RISC frente a uno CISC (*Complex Instruction Set Computer*) está basada en la mayor relevancia de este tipo de arquitectura. También cabe destacar que estos procesadores presentan una estructura y un repertorio de instrucciones fácil de comprender. Usando una ar-

quitectura RISC, los estudiantes aprenden los fundamentos básicos del repertorio de instrucciones y de la programación en lenguaje ensamblador en menos tiempo que utilizando un procesador CISC.

Simula3MS es un simulador de un procesador configurable, que cuenta con un entorno de trabajo sencillo que permite depurar fácilmente los programas, observar la evolución de la memoria, así como la ejecución de las instrucciones sobre distintos caminos de datos como pueden ser monociclo, multiciclo y segmentado. La interacción del usuario con la herramienta se hace por medio de una interfaz gráfica implementada con Java [8, 9].

Otro punto importante consiste en poder observar la evolución de todos estos componentes ciclo a ciclo (incluso paso a paso en el caso del procesador multiciclo) y tener también la opción de ejecutar conjuntamente todas las instrucciones y observar únicamente el efecto que produce la ejecución completa.

Consta de una primera parte que incluye un editor y que permite cargar un programa ya existente en un fichero o editarlo desde cero en la propia herramienta. En esta herramienta se analizan sintácticamente las instrucciones antes de pasar a la ejecución de las mismas. En esta parte también se permite escoger entre las distintas opciones disponibles para la configuración de los distintos caminos de datos.

Una vez analizadas sintácticamente las instrucciones se puede seguir la evolución del segmento de datos, así como de los registros y del resto del camino de datos.

3.1.1. Editor

La primera ventana que vemos al abrir *Simula3MS* (figura 1) es un editor.

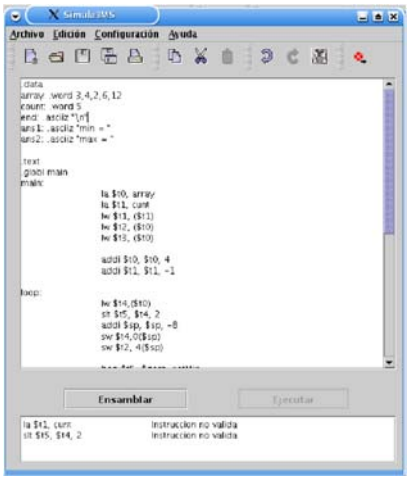


Figura 1: Ventana del editor.

La parte superior de esta ventana, engloba los menús y la barra de botones típicas de un editor, con la salvedad del menú *Configuración* y el botón de *Error siguiente*. El menú *Configuración* permite elegir al usuario entre los diferentes procesadores implementados (por defecto está seleccionado monociclo). El botón *Error siguiente* se activará en el caso de que una vez analizado el código se produzca más de un error sintáctico. Su función es ayudar al usuario en la corrección del código permitiendo avanzar al siguiente error de forma sencilla.

La zona central es un área de texto destinada a la elaboración del código en lenguaje ensamblador, y la zona inferior será utilizada para destacar los errores que se detecten al ensamblar el código.

Los botones, Ensamblar y Ejecutar aparecen desactivados en un principio. La finalidad de Ensamblar es indicar si el código es, o no, sintácticamente correcto; si lo es, se activará el botón Ejecutar, y si no se mostrarán los errores en la parte inferior de la pantalla, indicando brevemente el motivo. Al pulsar Ejecutar se pasará a la ventana que muestra el simulador, ésta será ligeramente diferente dependiendo de la configuración elegida.

3.1.2. Ventana de ejecución

Esta ventana presenta pequeñas variaciones según el tipo de camino de datos que se escoja. La figura 2 se corresponde con un procesador monociclo.

Los **registros** se muestran en la parte superior izquierda y están divididos en tres partes: *Registros especiales* (como son **PC**, **HI**, **LO**...), *Registros generales*, y los *Registros de punto flotante*.

En la parte superior derecha se puede ver el **camino de datos** correspondiente a un procesador monociclo sobre el que se representará la realización concreta de cada instrucción. En la implementación monociclo (figura 3.a) y segmentado (figura 3.b) el color de esta representación dependerá del tipo de

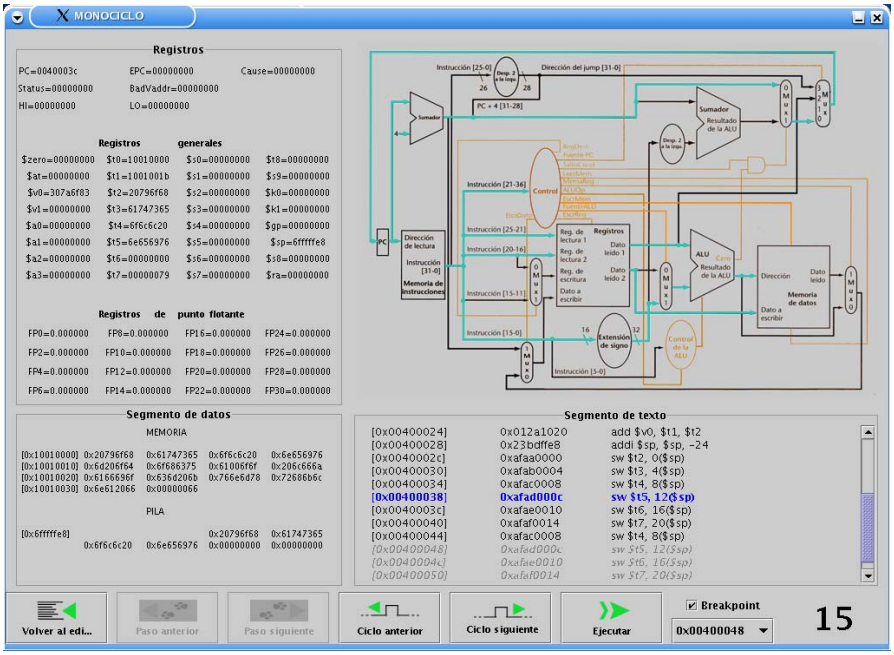


Figura 2: Ventana del simulador para un procesador monociclo.

la instrucción. Mientras, en el caso del multiciclo variará dependiendo de la etapa en la que nos encontremos.

La realización segmentada incluye además el *diagrama multiciclo*, figura 4, que se utiliza para dar una perspectiva general de diferentes situaciones dentro de la segmentación. Se considera que el tiempo avanza de izquierda a derecha, situación que se indica con el número de ciclo en la parte superior de la imagen, y las instrucciones se desplazan de la parte superior a la inferior del panel.

El **segmento de datos** se puede dividir en dos zonas: memoria y pila.

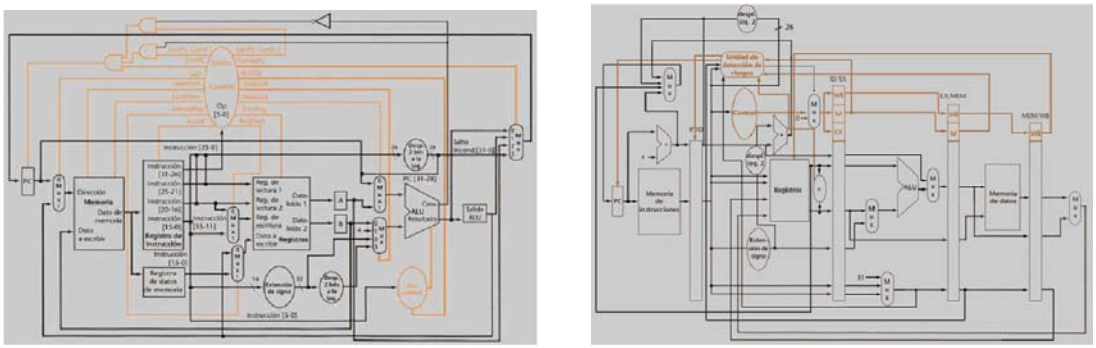
La memoria está dividida en dos partes, la primera columna, cuyo valor está entre corchetes es el valor de comienzo de la primera palabra de la línea. Las otras cuatro columnas son los datos almacenados en la memoria de forma consecutiva.

La *pila* crece hacia direcciones inferiores de memoria. Igual que la memoria esta zona también está dividida en dos partes, la primera columna indica la posición del puntero de pila y las restantes los datos.

El **segmento de texto** contiene una lista de instrucciones que componen el programa a ejecutar. Está dividido en tres columnas. En la primera columna se indica entre corchetes la dirección de memoria hexadecimal (**PC**) de la instrucción. Las instrucciones se almacenan a partir de la dirección 0x0040000. La segunda columna es la codificación numérica de la instrucción en hexadecimal, y la tercera columna es la instrucción en lenguaje ensamblador.

Aquella instrucción que esté siendo ejecutado en ese momento aparecerá resaltada en color azul. Si se ha insertado algún punto de ruptura (*breakpoint*) las instrucciones posteriores aparecerán en gris y en cursiva.

La parte inferior de la ventana contiene los siguientes botones: **Volver al editor**, **Ciclo anterior**, **Ciclo siguiente** y **Ejecutar** que ejecuta la totalidad del código o hasta el punto de ruptura. Además, en



(a) (b)

Figura 3: Caminos de datos (a) multiciclo (b) segmentado

la configuración multiciclo aparecen habilitados los botones de **Paso siguiente** y **Paso anterior**.

Después de estos botones hay un botón de selección (**Breakpoint**) que permite poner un punto de ruptura en el código. Si se selecciona este botón se activará el menú desplegable situado debajo de él para permitir elegir la situación del punto de ruptura entre las direcciones de las instrucciones desde la que se está ejecutando hasta la última del código. Por último, al final de la barra hay un número que indica el número de ciclos que han sido ejecutados hasta ese momento.

4. Experiencia de uso

Simula3MS se ha usado durante este curso en la asignatura **Estructura de Computadores I** [7] en las titulaciones de Ingeniería Informática e Ingeniería Técnica en Informática de Sistemas en la Universidad de A Coruña. Esta asignatura no se imparte en Ingeniería Técnica en Informática de Gestión. **Estructura de Computadores I** es una asignatura cuatrimestral de 7,5 créditos (6 teóricos y 1,5 prácticos) que se imparte en el primer cuatrimestre del segundo curso. Los alumnos que llegan a esta asignatura han cursado en el segundo cuatrimestre del primer curso la asignatura de **Tecnología de computadores** donde se estudian los circuitos lógicos básicos, tanto combinacionales como secuenciales. En el tercer curso los alumnos completarán sus conocimientos básicos de la arquitectura clásica con **Estructura de Computadores II**.

El objetivo de **Estructura de Computadores I** es introducir y analizar los principios básicos de la organización de los computadores. En esta asignatura se hace énfasis, principalmente, en los aspectos relacionados con el procesador.

Las prácticas en esta asignatura se dividen en sesiones de una hora semanales. Es obligatoria la asistencia de los alumnos al 80 % de las sesiones para poder superar la parte práctica de la asignatura. Cada sesión está dedicada a un tema concreto, como pueden ser los saltos condicionales, el paso de parámetros en subrutinas, el uso de la pila, etc. En cada una de las sesiones se entrega a los alumnos un guión de prácticas donde se detallan las instrucciones de la práctica y se indica claramente el objetivo de la misma. Además, estos guiones constan de una serie de cuestiones cortas que deben ir respondiendo durante el desarrollo de la práctica y entregar al profesor al finalizar la sesión. Este método fomenta en los alumnos una actitud más activa en el laboratorio, y permite que saquen mayor partido de cada sesión.

La acogida por parte de los alumnos ha sido muy buena, en especial por parte de aquellos que ya tenían experiencia con otros simuladores de años anteriores. Además, hemos constatado que en el examen de prácticas han obtenido mejores resultados que en los cursos precedentes. También hemos involucrado de forma indirecta a los alumnos en el

proceso de testeo de la herramienta, lo cual los ha motivado de cara a su uso.

5. Conclusiones y trabajo futuro

La principal utilidad de *Simula3MS* es su uso pedagógico por lo que el diseño y la implementación de la herramienta ha intentado paliar los defectos detectados en otras herramientas similares como **Spim** o **DLX**, que se venían usando hasta ahora.

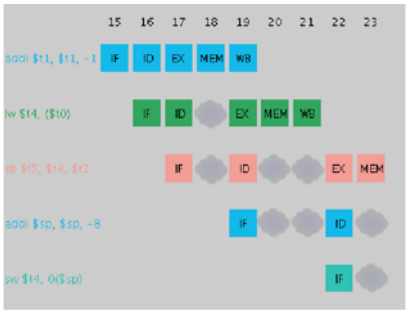


Figura 4: Diagrama multiciclo

Simula3MS se caracteriza por dividirse en dos partes. La primera de ellas, novedosa con respecto a las herramientas anteriores, se corresponde a un editor en el cual se analiza la corrección sintáctica del programa a ejecutar. En la segunda parte se visualiza el comportamiento del procesador que el usuario puede elegir en la configuración, y la evolución de los registros, la pila y la memoria durante la ejecución del programa. Las novedades en esta segunda parte son: la posibilidad de elegir el tipo de procesador, y la visualización del camino de datos y sección de control. *Simula3MS* permite la configuración de tres caminos diferentes: monociclo, multiciclo y segmentado.

Como trabajo futuro se aumentarán las prestaciones de *Simula3MS* incluyendo, por ejemplo, una unidad de punto flotante o distintas técnicas de segmentación como los algoritmos de Tomasulo y Marcador.

La herramienta está disponible para su uso libre en el enlace `http://www.des.udc.es/~patricia/-Simula3MS.htm`.

Referencias

[1] Simupro. http://www33.brinkster.com/vlaye-/software/simuproc/simuproc.html.

[2] Todd M. Austin. A Hacker's Guide to the Simplescalar Archiecture Research Tool Set. Technical report, Intel Micro Computer Research Labs, 1996.

[3] Erich Boehm. DLX Distribution Homepage. http://www.wu-wien.ac.at/usr/h93/h9301726/dlx.html, 1996.

[4] Robert L. Britton. *MIPS Assembly Languaje Programming*. Prentice Hall, 2004.

[5] Doug Burger and Todd M. Austin. The Simplescalar Tool Set, version 2.0. Technical report, Computer Science Departament, Univ. of Wisconsin, 1997.

[6] Raquel Concheiro. *Simula3MS*: Simulador de un procesador RISC multiciclo. Proyecto Fin de Carrera, Facultad de Informátic de la Univ. da Coruña, 2004.

[7] Estructura de Computadores I. http://www.des.udc.es/∼patricia/ec1.htm.

[8] Cay S. Horstmann and Gary Cornell. *Java 2. Características avanzadas*. Prentice Hall, 2003.

[9] Cay S. Horstmann and Gary Cornell. *Java 2. Fundamentos*. Prentice Hall, 2003.

[10] James R. Larus. SPIM A MIPS R2000/R3000 Simulator. http://www.cs.wisc.edu/-∼larus/spim.html.

[11] Marta Loureiro. *Simula3MS*: Simulador de un procesador RISC segmentado. Proyecto Fin de Carrera, Facultad de Informátic de la Univ. da Coruña, 2004.

[12] David A. Patterson and John L. Hennessy. *Estructura y diseño de computadores. Interficie circuitería/programación*. Reverté, S.A., 2000.

[13] Simula3MS. http://www.des.udc.es/∼patricia/-Simula3MS.htm.

[14] Fermín Sánchez. Características deseables en un procesador pedagógico parala enseñanza básica de la arquitectura de computadores. *Jornadas de Enseñanza Universitaria de la Informática (JENUI)*, 2002.

[15] William Stallings. *Organización y arquitectura de computadores*. Prentice Hall, 2000.

[16] Miguel A. Vega and Juan A. Gómez Juan M. Sánchez. Innovación docente en la Arquitectura de Computadores mediante el uso de Simuladores. *Jornadas de Enseñanza Universitaria de la Informática (JENUI)*, 2000.