Mejorando la docencia en las prácticas de la asignatura Arquitectura e Ingeniería de Computadores

Miguel A. Vega Rodríguez, Juan M. Sánchez Pérez, Juan A. Gómez Pulido
Universidad de Extremadura. Dpto. de Informática
Escuela Politécnica. Campus Universitario, s/n. 10071 Cáceres. Spain
Correo-e: {mavega, sanperez, jangomez}@unex.es. Fax: +34-927-257202

Resumen

En la actualidad es habitual la utilización de simuladores en las prácticas de muchas de las asignaturas de Informática. El uso de estos simuladores facilita a los alumnos la comprensión de un gran número de conceptos prácticos que de otra forma, y a pesar de la teoría impartida, quedarían algo difusos. En esta ponencia se presentan las mejoras docentes aplicadas en las prácticas de la asignatura Arquitectura e Ingeniería de Computadores, con el ánimo de darlas a conocer a otros profesores, y buscando siempre el intercambio de ideas y experiencias. En estas prácticas, se utiliza un nuevo enfoque: los alumnos deben construir versiones reducidas de diversos simuladores, y no sólo usarlos. Dicho de otro modo, los alumnos cambian el papel de usuarios por otro más activo, el de desarrolladores de estos recursos didácticos. En este trabajo se detallan varias de las prácticas (simuladores) realizadas por los estudiantes, indicando su funcionalidad, dificultad,... En conclusión, se ha comprobado que la construcción de estos simuladores obliga a los estudiantes a conocer con más profundidad las consideraciones teóricas en las que se apoyan, que si sólo los usaran, por lo que adquieren un conocimiento mejor.

1. Introducción

En la enseñanza universitaria de la Informática, la realización de prácticas es un aspecto primordial para la comprensión de muchos de los conceptos impartidos. Hoy en día, es habitual que estas prácticas conlleven, por parte del alumnado, el uso de simuladores. Simuladores desarrollados, en la mayoría de ocasiones, por los profesores que imparten las asignaturas. De hecho, en la literatura puede encontrarse un gran número de publicaciones en las que se detallan nuevos recursos didácticos desarrollados por profesores para ser utilizados por estudiantes ([5], [1] y [7] son algunos ejemplos dentro de las JENUI).

Nuestra ponencia busca un enfoque diferente. En ella defendemos la idea de que sean los propios alumnos los que desarrollen estos simuladores en las clases prácticas. Cambiamos el papel del alumno, por otro más activo, de usuario a desarrollador de estas herramientas. Lógicamente, adecuando siempre la dificultad del simulador a desarrollar a la capacidad de los alumnos. Se trata pues de un enfoque distinto al encontrado en la literatura.

En este trabajo describimos varios de los simuladores (prácticas) realizados por los propios estudiantes. Siguiendo el principio de "vale más una imagen que mil palabras", para todas estas prácticas incluimos varias imágenes de los resultados obtenidos por los alumnos. De esta forma, con un simple vistazo, el lector podrá observar la funcionalidad, dificultad, temática, ... de las prácticas propuestas.

El resto del artículo se organiza como sigue: en la próxima sección describimos brevemente el contexto en el que se realizan estas prácticas. La sección 3 explica varias de las prácticas propuestas a los alumnos, mostrando múltiples imágenes de los resultados conseguidos, sus datos más significativos, etc. En la sección 4 se analizan los resultados obtenidos de la aplicación de este nuevo enfoque a las prácticas. Finalmente, en la última sección se presentan las conclusiones de este trabajo.

2. Contexto

Las prácticas aquí descritas se realizan dentro de la asignatura Arquitectura e Ingeniería de Computadores (AIC). Una asignatura troncal anual de 9 créditos, que se imparte en cuarto curso de la titulación de Ingeniero en Informática, dentro de la Universidad de Extremadura (UEx). Esta asignatura tiene como objetivo general

formar al alumno en los fundamentos arquitectónicos de los computadores actuales. Este objetivo abarca el estudio de muy diversos campos: circuitos aritméticos complejos, segmentación aritmética y de instrucciones, multiprocesadores, multicomputadores, etc. Para obtener información más detallada sobre el programa de la asignatura el lector puede consultar la referencia [8].

3. Prácticas desarrolladas

En los siguientes apartados se presentan varias de las prácticas propuestas a los alumnos en AIC. Para todas ellas se da una breve descripción de la funcionalidad, así como múltiples imágenes del aspecto final del simulador entregado, en cada caso, por los alumnos. Como regla general, los alumnos realizan estas prácticas en grupos de dos, aunque también hay prácticas que han sido desarrolladas por un único alumno. Cada grupo de prácticas puede escoger una determinada temática para desarrollar su simulador o incluso proponer una distinta a las ya existentes. Cada grupo debe implementar uno de estos simuladores (al completo) durante un cuatrimestre. Para facilitar su labor se les permite elegir el entorno de programación a usar. El otro cuatrimestre se dedica a realizar múltiples prácticas que cubren la mayor parte del temario teórico.

El objetivo buscado en esta sección no es ser completos, pues, por motivos de espacio, es imposible detallar todas las prácticas propuestas así como los resultados conseguidos por todos los alumnos. En cualquier caso, se ha buscado resaltar las prácticas (simuladores) más significativas, además del resultado conseguido para cada una de ellas por algún grupo. De esta forma, el lector puede obtener ideas para posibles prácticas en su propia asignatura, además de evaluar la conveniencia de este enfoque en sus prácticas.

3.1. Simulador del MIPS R2000

Esta práctica simula el comportamiento de la máquina MIPS R2000 segmentada, siguiendo la notación y diagramas indicados en [2]. Además se exigió que el simulador funcionara vía Internet.

Presentamos el simulador implementado por uno de los grupos de prácticas. La aplicación desarrollada se realizó en Java mediante JBuilder, poseyendo una interfaz gráfica completa (ver figura 1), además de permitir el uso tanto del navegador Internet Explorer como Netscape.

Como se muestra en la figura 1, la información en pantalla cambia continuamente según se avanza en la simulación. Estos cambios se pueden desglosar en cuatro zonas (figura 1):

a) Zona donde se muestra la instrucción que se está ejecutando en cada etapa. A cada instrucción se le asigna un color para facilitar su seguimiento por las distintas etapas.

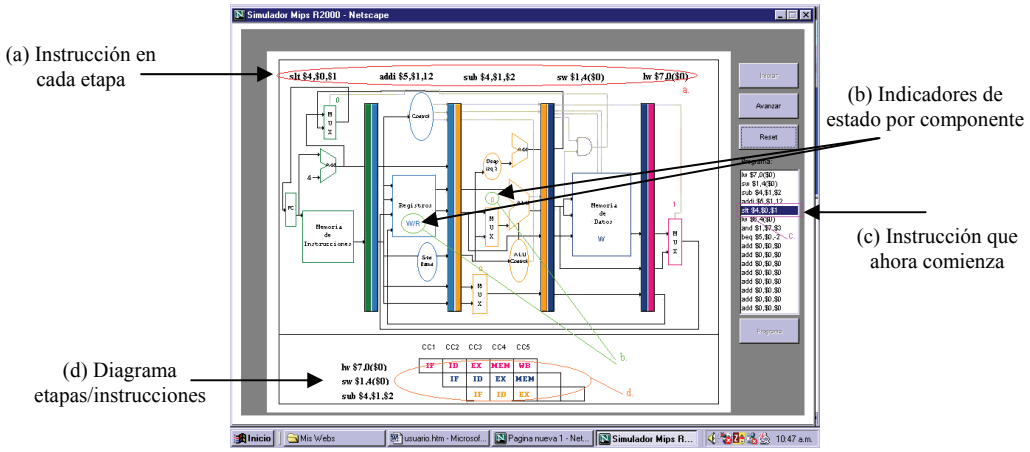

Figura 1. Interfaz del simulador del MIPS R2000

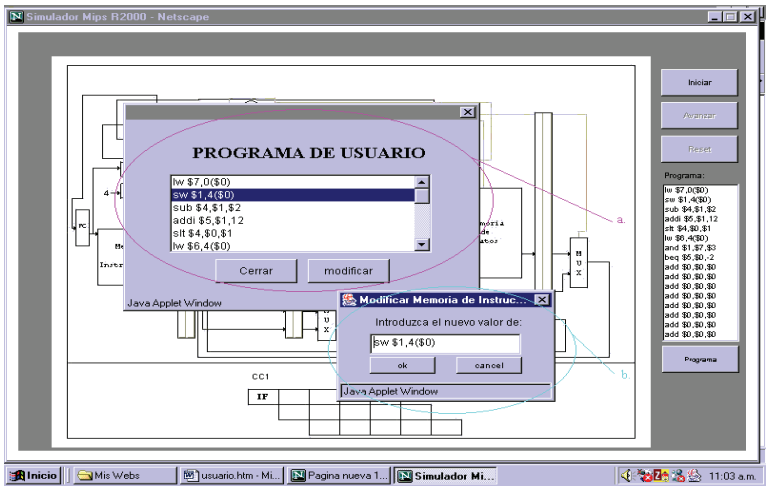

Figura 2. Modificación de la memoria de instrucciones (programa a simular) del MIPS R2000

b) Indicadores de estado para cada componente de la máquina. En los multiplexores se indica su entrada de control, y en la memoria de datos y el banco de registros se especifican las operaciones de lectura y escritura. No se muestra ningún indicador para la memoria de instrucciones, puesto que siempre se leerá una nueva instrucción.

c) Indicador de la instrucción que empieza su ejecución.

d) En la parte inferior de la pantalla se muestra un diagrama para los últimos 7 ciclos de reloj, indicando en qué etapa está cada instrucción. En este diagrama inicialmente aparecen las tres primeras instrucciones, y según vayan finalizando se mostrará el estado de las instrucciones siguientes.

Para comenzar la simulación se pulsa el botón *Iniciar*. Esto hará que comience a ejecutarse la primera instrucción en la etapa 1 (IF). Para avanzar un ciclo de reloj en la simulación se pulsa el botón *Avanzar*. Si se desea interrumpir la simulación se utiliza el botón *Reset* que situará la máquina en su estado inicial.

Durante la simulación podemos ver y modificar el contenido del banco de registros, de la memoria de instrucciones, y de la memoria de datos. Para ello haremos clic con el ratón sobre cualquiera de estos componentes. De esta forma aparecerá una ventana que nos permitirá ver su estado actual y modificarlo. En la figura 2 se muestra un ejemplo para la memoria de instrucciones. En este caso, si la instrucción introducida no es reconocida por el analizador sintáctico se mostrará un mensaje de error, indicando la sintaxis de la instrucción.

3.2. Método del marcador

En esta práctica se ha desarrollado una aplicación que simula la planificación dinámica de instrucciones, utilizando un subconjunto de instrucciones de la máquina *DLX* y el método del marcador (*Scoreboarding*) [3]. Presentamos aquí la práctica realizada por uno de los grupos.

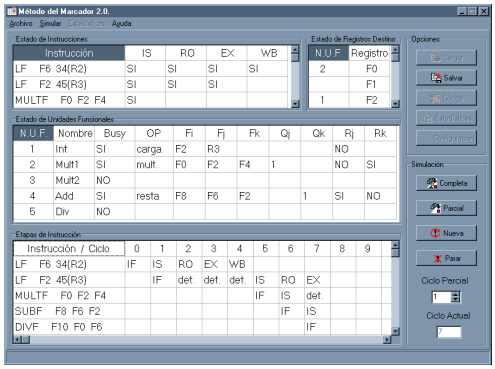

Figura 3. Interfaz del simulador para el marcador

El programa, desarrollado en Delphi, utiliza un entorno de ventanas (*Windows*) donde se muestran

las cuatro tablas que contienen la información del marcador e indican el estado de las instrucciones, las unidades funcionales, los registros destino y la evolución de las distintas etapas de las instrucciones (ver figura 3).

La simulación puede ser de dos tipos: total, que realiza una simulación completa del código; o parcial, que simula hasta un número determinado de ciclos de reloj, que indica el usuario. Los resultados de la simulación pueden ser guardados en un fichero (con extensión *mcd*), para posteriormente, abrir ese fichero y continuar una simulación guardada en algún punto de su desarrollo.

Para ver los riesgos que se han producido durante la planificación, la aplicación muestra una estadística, en la que se presenta un gráfico de tipo "tarta" (figura 4), con el número de riesgos que se han producido en la ejecución del código. En este gráfico, cada color se corresponde con un tipo de riesgo, según la leyenda de la parte superior derecha de la figura 4.

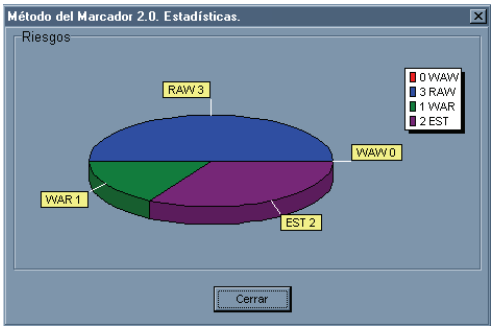

Figura 4. Estadísticas de riesgos en el marcador

El simulador también posee un *Editor de Código* que permite crear y modificar el código de los programas que se quieren simular. Para construir las instrucciones, el editor posee una serie de campos donde se seleccionan la instrucción y sus registros fuentes y destino, simplemente pulsando con el puntero del ratón sobre ellos (figura 5). Si la instrucción no está formada correctamente, se informa de este hecho al usuario mediante un mensaje por pantalla. El código editado puede guardarse en un fichero (*.dlx), pudiéndose recuperar en otro momento (abriendo dicho fichero para cargar su código en el editor).

Figura 5. Editor de código del marcador

Además, el simulador permite al usuario modificar el número de ciclos de reloj que tardará en ejecutarse la etapa *EX* en las distintas unidades funcionales. El máximo valor para todas las unidades funcionales es de 999 ciclos de reloj, y el valor mínimo es de 1 ciclo de reloj.

Finalmente, destacar que la aplicación también posee sistema de ayuda y es autoinstalable.

3.3. Aritmética en la web

En esta práctica se ha desarrollado una aplicación que simula el comportamiento de cuatro circuitos aritméticos complejos, dos multiplicadores y dos divisores. En concreto, los multiplicadores seleccionados son: multiplicador de Booth y multiplicador de Baugh-Wooley. Mientras que los algoritmos divisores son los siguientes: divisor sin restauración y divisor con restauración [6].

La práctica que aquí presentamos se desarrolló mediante Macromedia Flash. La elección de Flash para la implementación de esta aplicación se fundamentó en el uso del simulador a través de Internet, y en que Flash facilita el diseño de entornos gráficos con animación.

Los dos multiplicadores han sido representados mediante su circuito, mientras en los divisores lo que se representa es el algoritmo seguido por éstos, puesto que este algoritmo detalla con más claridad el funcionamiento de cada divisor.

Por ejemplo, en la figura 6 se muestra la interfaz para el multiplicador de Booth. En ella podemos observar tres cuadros de texto en los que se indicará el valor del primer y segundo operando a multiplicar, así como el número de bits que va a usar el circuito para sus datos.

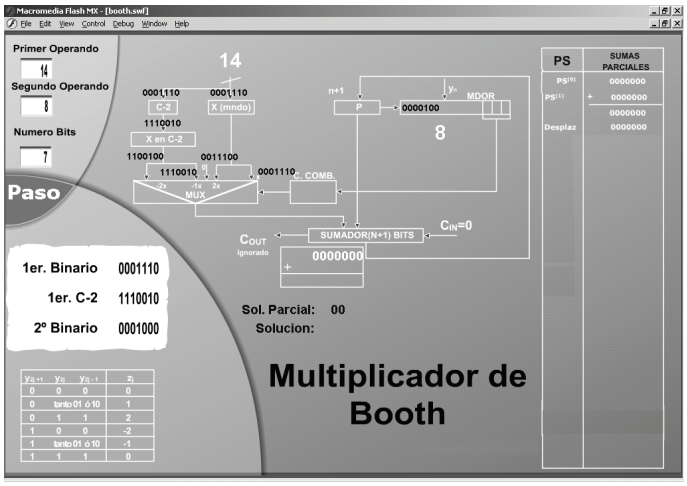

Figura 6. Interfaz para el multiplicador de Booth

Tras estos datos, con el botón *Paso* podemos ir avanzando en la simulación. Observamos que la tabla de la derecha se va rellenando a medida que se realiza la multiplicación. La tabla canónica, que está situada abajo a la izquierda, va mostrando en cada caso cuál es la selección del multiplexor.

Para los divisores, la figura 7 muestra un ejemplo. En esta figura se presenta el simulador para el divisor con restauración. El manejo básico de este simulador es similar al explicado para los multiplicadores. En primer lugar hay que introducir los valores para el dividendo, divisor y número de bits a usar en la operación. Una vez introducidos correctamente estos datos, pulsamos el botón *Bin* y el algoritmo comienza a simularse paso a paso.

La tabla que hay a la derecha del simulador se va rellenando de forma dinámica, dependiendo de los números introducidos en los cuadros de texto. Al terminar la simulación, podremos comprobar el resultado final de la división.

3.4. Algoritmo de Tomasulo

En esta práctica se ha desarrollado un simulador para la planificación dinámica de instrucciones utilizando el algoritmo de Tomasulo [3].

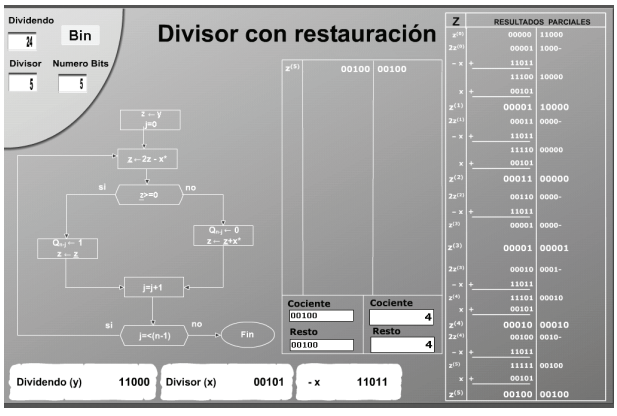

Figura 7. Interfaz para el divisor con restauración

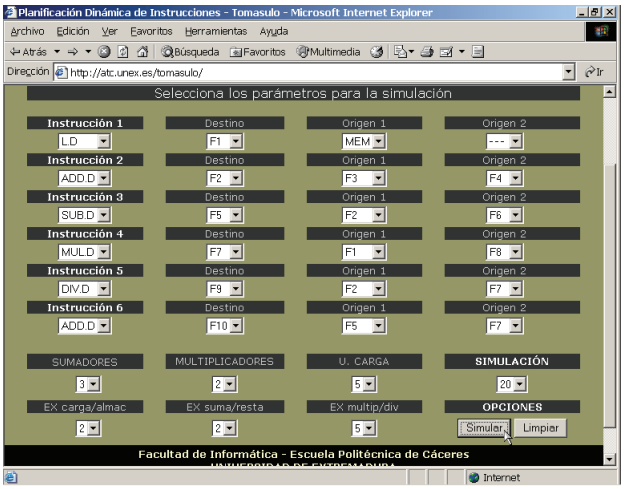

Figura 8. Interfaz para establecer los parámetros de simulación (algoritmo de Tomasulo)

En la práctica que mostramos aquí, el simulador se ha implementado para que trabaje vía Internet (sistema cliente/servidor), por lo que ha sido programado utilizando el lenguaje ASP, desarrollando secuencias de órdenes (*scripts*) junto con código HTML.

En este caso, el cliente establece los parámetros de simulación (figura 8) y envía su petición al servidor, que realiza los procesos necesarios y envía al cliente el resultado de la simulación en formato web (figura 9).

Los parámetros de configuración incluyen todos aquellos datos que son de interés para realizar la simulación: instrucciones, registros destinos, registros fuentes, cantidad de ciclos a mostrar, etc.

Por otro lado, como podemos observar en la figura 9, el simulador ofrece la posibilidad de navegar entre los distintos ciclos de la simulación. Para cada ciclo, el simulador muestra mediante tablas los datos necesarios para comprender el funcionamiento del algoritmo de Tomasulo, para una secuencia de instrucciones concreta.

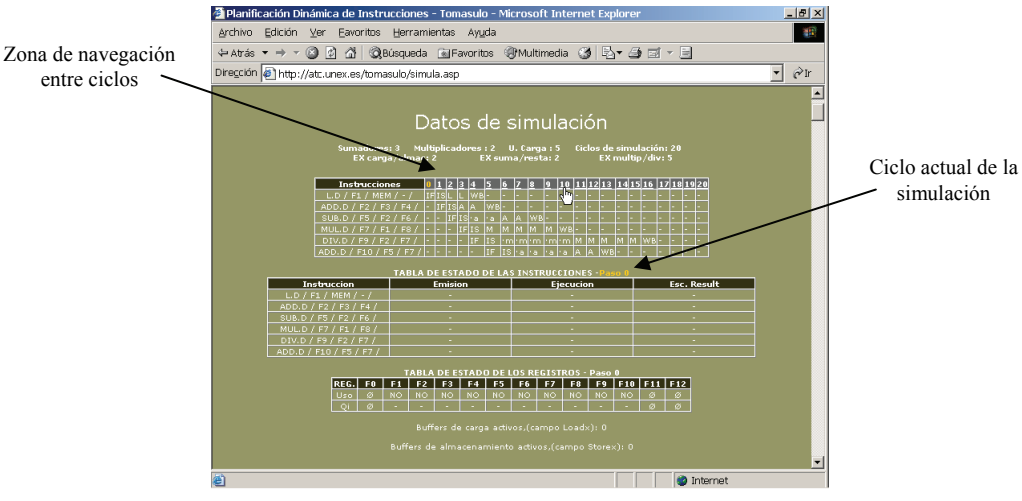

Figura 9. Interfaz para mostrar los resultados de la simulación (algoritmo de Tomasulo)

3.5. Simulador de redes multietapa

Con esta práctica se ha construido un simulador que permite ver la topología de las redes multietapa más comunes en multicomputadores [4]. Además de la topología, también se puede estudiar el encaminamiento para un determinado conjunto de conexiones, e incluso consultar la ruta aislada que un nodo de entrada tiene hacia un nodo de salida en base a un encaminamiento dado.

Figura 10. Selección del tipo de red multietapa

El simulador (la práctica), que aquí presentamos, se ha desarrollado para ser utilizado a través de Internet, habiéndose programado en Authorware de Macromedia.

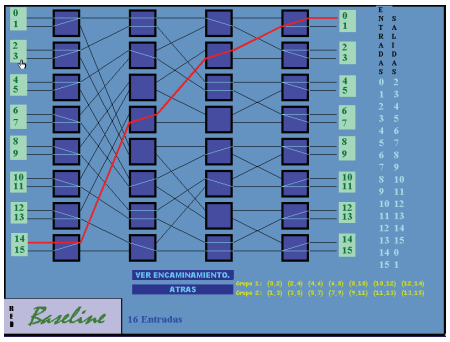

Figura 11. Encaminamiento en una red Baseline 16x16

La figura 10 muestra la pantalla en la que el usuario selecciona la red multietapa a estudiar. A esta pantalla siguen otras para delimitar el nº de entradas-salidas, etc. Tras estas pantallas, aparece el aspecto de la red seleccionada (ver figura 11 para la red Baseline 16x16). Mostrada la topología de dicha red, se puede estudiar el encaminamiento para un conjunto de conexiones dado o incluso calcular la solución a los bloqueos existentes (agrupación de las conexiones en varios pasos), en el caso de que se trate de una red con bloqueo.

La figura 11 también presenta las conexiones (entrada-salida) configuradas por el usuario (lateral derecho), y la ruta a seguir para conectar, en este caso, la entrada 14 con la salida 0.

3.6. Simulador de memorias caché en SMPs

En esta práctica los alumnos han desarrollado una versión muy reducida del simulador SMPCache [9]. La aplicación desarrollada simula un sistema de memoria jerárquico (basado en dos niveles: memoria principal y caché) en un SMP (*Symmetric MultiProcessor*, Multiprocesador Simétrico) con memoria compartida por bus.

El mini-simulador desarrollado permite la configuración de múltiples parámetros: número de procesadores, protocolo de coherencia, arbitración del bus, tamaño de la caché, de la memoria principal, del bloque de memoria, tipo de correspondencia, reemplazo, etc.

Además, presenta en pantalla una gran cantidad de datos de interés: transacciones en el bus, transferencias de bloques, transiciones de estado (de los bloques de caché), accesos a memoria realizados, y desglose por tipos, tasa de fallos y aciertos,... La figura 12 muestra un ejemplo de las prácticas realizadas.

4. Resultados

En esta sección intentamos evaluar la calidad de las mejoras introducidas. Para ello, resumimos las principales ventajas e inconvenientes del nuevo enfoque utilizado en prácticas.

Entre las ventajas destaca el hecho de que se ha comprobado que la implementación de estos simuladores, durante un cuatrimestre, obliga a los alumnos a conocer en profundidad y dominar las consideraciones teóricas en las que se apoyan (más que si sólo los usaran), lo que produce una mejora en la calidad de la enseñanza de esa parte del temario. Además, no debe olvidarse que durante el otro cuatrimestre se realizan múltiples prácticas que cubren la mayor parte de la teoría. Completando así la educación de los estudiantes.

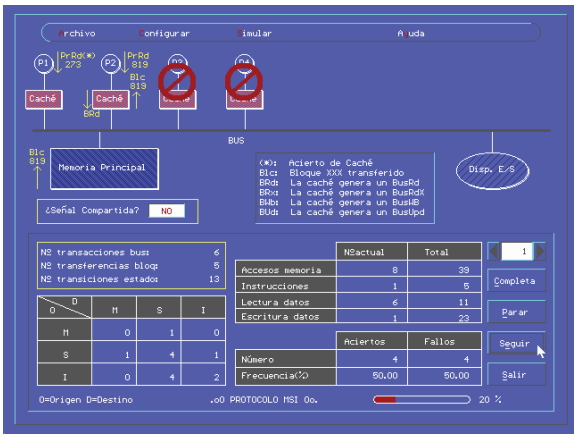

Figura 12. Versión reducida del simulador SMPCache, realizada por los alumnos en prácticas

Respecto a los inconvenientes, el principal punto en contra es el trabajo extra que este enfoque requiere por parte del profesor:

- Debe proponer diversos simuladores a ser implementados por los alumnos, preparando la documentación necesaria.
- Obliga, al profesor de prácticas, a tener un conocimiento muy profundo sobre cualquier parte del temario teórico que pueda usarse para realizar un simulador.
- Debe realizar el seguimiento (tutorías,...) y evaluación de prácticas muy diversas: en entornos de programación, temáticas, etc.

5. Conclusión

En este trabajo se han indicado los cambios docentes que se han introducido en las prácticas de la asignatura AIC. Estos cambios se fundamentan en la implementación completa de un simulador (con temática a elegir) por parte de los alumnos durante un cuatrimestre, mientras dedican el otro cuatrimestre a varias prácticas sobre todo el temario. Pensamos que esta combinación (prácticas variadas más pequeñas + 1 práctica/simulador más grande y en detalle) es muy interesante y útil a nivel docente.

Esta innovación en el enfoque de las prácticas ha sido bien aceptada por los alumnos, habiéndose detectado las ventajas e inconvenientes explicados en la sección 4. En conclusión, las ventajas producidas son tan importantes que hacen razonable el esfuerzo extra por parte del profesor. Además, quizás sea ésta una forma de acometer, en parte, la adaptación de esta asignatura al EEES, donde se demanda un papel más activo y protagonista por parte del estudiante.

Finalmente, indicar que esta propuesta también puede usarse en otras muchas asignaturas, donde el uso de simuladores sea aconsejable.

Referencias

[1] Almisas, R.; Paz, R.; Linares, A.; Amaya, C.; Sevillano, J.L. *Un simulador de memorias caché multinivel*. JENUI 2001, pp. 429-432.

[2] Hennessy, J.L.; Patterson, D.A. *Organización y diseño de computadores: la interfaz hardware/software.* McGraw-Hill, 1999.

[3] Hennessy, J.L.; Patterson, D.A. *Computer architecture: a quantitative approach.* Morgan Kaufmann, 2003.

[4] Hwang, K. *Advanced computer architecture: parallelism, scalability, programmability*. McGraw-Hill, 1993.

[5] Martínez, J.C.; López, P. *Simulador de un computador segmentado con especulación hardware*. JENUI 2000, pp. 53-59.

[6] Parhami, B. *Computer arithmetic: algorithms and HW designs*. Oxford Univ. Press, 2000.

[7] Prieto, M.; de Vicente, A.J.; Vargas, J.A. *Simulador de dispositivos de entrada/salida programables*. JENUI 2002, pp. 595-598.

[8] Vega, M.A., Sánchez, J.M., Ballesteros, J. http://arco.unex.es/mavega/AIC.htm, 2006.

[9] Vega, M.A. http://arco.unex.es/smpcache, 2006.