Utilización del emulador em88110 para la realización de prácticas de las asignaturas del plan 96 de la Facultad de Informática de la U.P.M.

María Isabel García Clemente
Santiago Rodríguez de la Fuente
Rafael Méndez Cavanillas
Departamento de Arquitectura y Tecnología de Sistemas Informáticos
Universidad Politécnica de Madrid
Email: mgarcia@fi.upm.es, srodri@fi.upm.es, rmendez@fi.upm.es

Resumen

La introducción del nuevo plan de estudios de la Facultad de Informática de la U.P.M. con la correspondiente reestructuración de las asignaturas que cubren los conocimientos de Estructura y Arquitectura de Computadores, ha hecho necesaria la reestructuración del contenido práctico de dichas asignaturas, en la que se ha considerado la necesidad de incluir prácticas en las que se contemplen conceptos considerados básicos en la actualidad.

Por otra parte, la realización de prácticas sobre una máquina real comercial (p.e. un SuperSparc) plantea una serie de inconvenientes que complican en exceso la asimilación de los conceptos que se desean comprender.

Aquí se plantea la utilización de un emulador que proporcione al alumno un entorno de trabajo integrado, que pueda ejecutarse en distintas plataformas (desde PC's hasta estaciones de trabajo multiprocesador) y permita la realización de distintas prácticas, de complejidad progresiva. Con ello perseguimos afianzar los conocimientos teóricos que se deben adquirir, al cursar las asignaturas de Estructura, Laboratorio y Arquitectura de Computadores, las dos primeras correspondientes a segundo curso y la tercera a tercer curso.

1 Introducción

La puesta en marcha del nuevo plan de estudios de Ingeniería Informática en la Facultad de Informática de la U.P.M. ha hecho necesaria la reestructuración del contenido práctico de las asignaturas que cubren los temas básicos de Estructura y Arquitectura de Computadores.

En este nuevo plan de estudios se ha producido una disminución de aproximadamente un 10% de los créditos teóricos y un incremento aproximado de un 75% de los créditos prácticos, con respecto al anterior plan de estudios del 83.

Por otra parte, materias como procesadores superescalares o mecanismos de transformación de programas para aumentar las prestaciones de distintos componentes del computador, que antes se trataban en asignaturas optativas, o incluso en seminarios, en la actualidad se deben considerar materias que todo Ingeniero en Informática debe conocer, por lo que creemos necesario incluirlas como obligatorias.

Por todo ello, nuestra propuesta ha sido descargar de contenido teórico algunos temas como la programación en Ensamblador, el Sistema de Memoria y los computadores de altas prestaciones, entre otros, de modo que estos aspectos se traten como parte de una serie de prácticas obligatorias. Es con la realización de prácticas con lo que realmente el alumno asimila los conceptos teóricos que les presentamos.

Para la realización de estas prácticas proponemos la utilización de emuladores, dada la complejidad añadida que supone el utilizar computadores comerciales, que en algunos casos no hacen más que desviar la atención del alumno a otros aspectos que nada tienen que ver con los objetivos que se desean conseguir. Además consideramos más positivo utilizar un mismo sistema para realizar distintas prácticas que utilizar uno distinto para cada una de ellas, puesto que está comprobado que una de las tareas qué lleva más tiempo en la realización de una práctica es la comprensión del sistema que sirve de soporte.

En nuestro caso proponemos la utilización de un emulador del procesador Motorola 88110. Si bien se trata de un procesador superescalar, el emulador está diseñado de forma que permite ser configurado para ejecutar programas de forma totalmente secuencial, sin tener en cuenta la existencia de *pipeline*, memorias cache, ni la característica de superescalar, o bien ser configurado para hacer visibles al alumno todas estas características. Asimismo permite obtener algunas estadísticas internas del computador emulado.

Este enfoque permite realizar prácticas de complejidad creciente, desde la ejecución secuencial de un programa escrito en lenguaje Ensamblador y su depuración, pasando por la configuración de los distintos parámetros de las memorias cache para evaluar la configuración más adecuada para un programa particular, hasta el tratamiento de aspectos más complejos como el *pipeline*, el procesamiento superescalar y las dependencias entre instrucciones, así como el efecto que producen éstas sobre el rendimiento del procesador.

Actualmente existen dos versiones operativas del emulador, cuya única diferencia estriba en el tipo de interfaz que ofrecen, que en un caso es el clásico emulador de líneas de comandos, mientras que en el otro se trata de un interfaz basado en X-window.

Las prácticas que se proponen, utilizando dicho emulador, son las que se describen a continuación.

2 Programación en lenguaje Ensamblador

Esta práctica corresponde a la asignatura Laboratorio de Estructura de Computadores, que se imparte en el segundo cuatrimestre de segundo curso, y cuyo contenido es eminentemente práctico (el 75% de los créditos son prácticos).

El objetivo principal de esta práctica es que el alumno se familiarice con la arquitectura de un procesador, a través del conocimiento y utilización de su juego de instrucciones. Hay que tener en cuenta que ésta es la primera vez, a lo largo de la carrera, en la que el alumno tiene la posibilidad de conocer las posibilidades que proporciona un procesador al programador, así como sus limitaciones.

Adicionalmente, se pretende que el alumno ponga en práctica algunos conceptos teóricos relacionados con la programación (subrutinas, paso de parámetros, etc.) así como consolidar conceptos estudiados en la parte teórica de la asignatura previa de Estructura de Computadores, en particular conceptos correspondientes a los temas de Instrucciones y Direccionamientos, Representación y Aritmética del Computador.

La práctica que proponemos consiste en la programación, en ensamblador del procesador Motorola 88110, de un algoritmo que permita realizar una determinada operación (p.e. una suma) sobre números representados en un determinado formato de coma flotante y realizar la conversión del resultado a otro formato de coma flotante distinto o a un formato de representación de enteros.

El algoritmo deberá estructurarse como un programa principal y al menos dos subrutinas que realicen la operación de suma en coma flotante y la conversión del resultado respectivamente. Esta restricción tiene como objetivo que el alumno esté obligado a programar el paso de parámetros y a gestionar la llamada y retorno de subrutina, utilizando los mecanismos que le proporciona la arquitectura del procesador. En este caso se propone que el paso de parámetros

así como la devolución de resultados se realice mediante el mecanismo tradicional, es decir a través de una pila.

Debido a que el procesador emulado no dispone de un registro de propósito específico para la gestión de la pila se propone asignar, para realizar esta labor, uno de los registros de propósito general, que hará las funciones del clásico *stack pointer*. Éste deberá apuntar a la dirección de la palabra que ocupa la cabecera de la pila (donde estará la última información introducida), que crecerá, como es habitual, hacia direcciones de memoria decrecientes.

El espacio asignado para las variables locales de cada subrutina se reservará en el marco de pila correspondiente a dicha subrutina. El alumno deberá construir y gestionar el acceso a dicho marco de pila, asignando a uno de los registros generales la función del clásico *frame pointer*.

Asimismo, se propone utilizar en esta práctica los dos métodos clásicos de paso de parámetros: por dirección y por valor, no permitiéndose utilizar las instrucciones de multiplicación y división entera ni las instrucciones de coma flotante de que dispone el procesador.

Por otra parte, y dado que las instrucciones de salto con retorno que proporciona el 88110 (**jsr** y **bsr**) salvaguardan la dirección de retorno en el registro **r1**, se propondrá al alumno que incluya un mecanismo que permita realizar llamadas anidadas a subrutinas, mediante la utilización de la pila anteriormente mencionada.

Para facilitar la labor de programación se dispone de un Ensamblador que permite generar el código binario ejecutable por el emulador. Este ensamblador, además de admitir las instrucciones propias del procesador, admite algunas pseudoinstrucciones especificadas en el estándar IEEE 694, y ya conocidas por el alumno, como son: **org**, **res** y **data**. Permite además realizar las referencias a las distintas variables de forma simbólica, es decir a través de sus nombres o etiquetas, e incorpora la posibilidad de definir y utilizar *macros*, para aquellos fragmentos de código que el alumno deba repetir con frecuencia.

Una vez obtenido el fichero ejecutable, el siguiente paso es ejecutar el programa. Para ello, se deberá configurar el emulador en modo serie, lo que supone, de forma implícita, la inhibición de las características del procesador que el alumno aún no conoce, tales como las memorias cache y el pipeline de instrucciones, haciendo transparente al alumno el hecho de que el computador que está utilizando es superescalar.

```
 PC=0          or       r01,r00,992     Tot. Inst: 0    << Ciclo : 1
 FL=1 FE=1 FC=0 FV=0 FR=0
 R01 = 00000000 h R02 = 00000000 h R03 = 00000000 h R04 = 00000000 h
 R05 = 00000000 h R06 = 00000000 h R07 = 00000000 h R08 = 00000000 h
 R09 = 00000000 h R10 = 00000000 h R11 = 00000000 h R12 = 00000000 h
 R13 = 00000000 h R14 = 00000000 h R15 = 00000000 h R16 = 00000000 h
 R17 = 00000000 h R18 = 00000000 h R19 = 00000000 h R20 = 00000000 h
 R21 = 00000000 h R22 = 00000000 h R23 = 00000000 h R24 = 00000000 h
 R25 = 00000000 h R26 = 00000000 h R27 = 00000000 h R28 = 00000000 h
 R29 = 00000000 h R30 = 00000000 h R31 = 00000000 h
88110> c
0 Serie, Little Endian,
Redondeo al mas proximo, Excepciones Inhibidas

Memoria:
        T. Acceso: 10 ciclos
        N. Bloques: 1
Cache de instrucciones: INHIBIDA
Cache de datos: INHIBIDA
```

Figura 1: Inicio del emulador en la versión de línea de comandos.

Otro de los parámetros que se pueden configurar son: el modo de redondeo a utilizar, permitiéndose seleccionar entre redondeo al más próximo, a cero, a -infinito y a +infinito, todos ellos estudiados en teoría, así como el orden de almacenamiento de los bytes en memoria (permitiéndose seleccionar *big-endian* o *little-endian*) y la habilitación o inhibición de las interrupciones.

En esta práctica se proporcionará al alumno una configuración fija para no distraer su atención con estas cuestiones, ya que lo que realmente tiene que hacer es realizar un programa en ensamblador, ser capaz de seguir su ejecución y hacer que ésta sea correcta.

Al invocar al emulador, pasándole como argumento el programa ya ensamblado, éste muestra el estado del procesador en ese momento. Las figuras 1 y 2 ilustran el formato en el que se presenta dicho estado para las dos versiones del emulador. En la figura 1 se muestra además el modo en el que está configurado el emulador, que se puede visualizar utilizando uno de sus comandos (en nuestro caso, **c**).

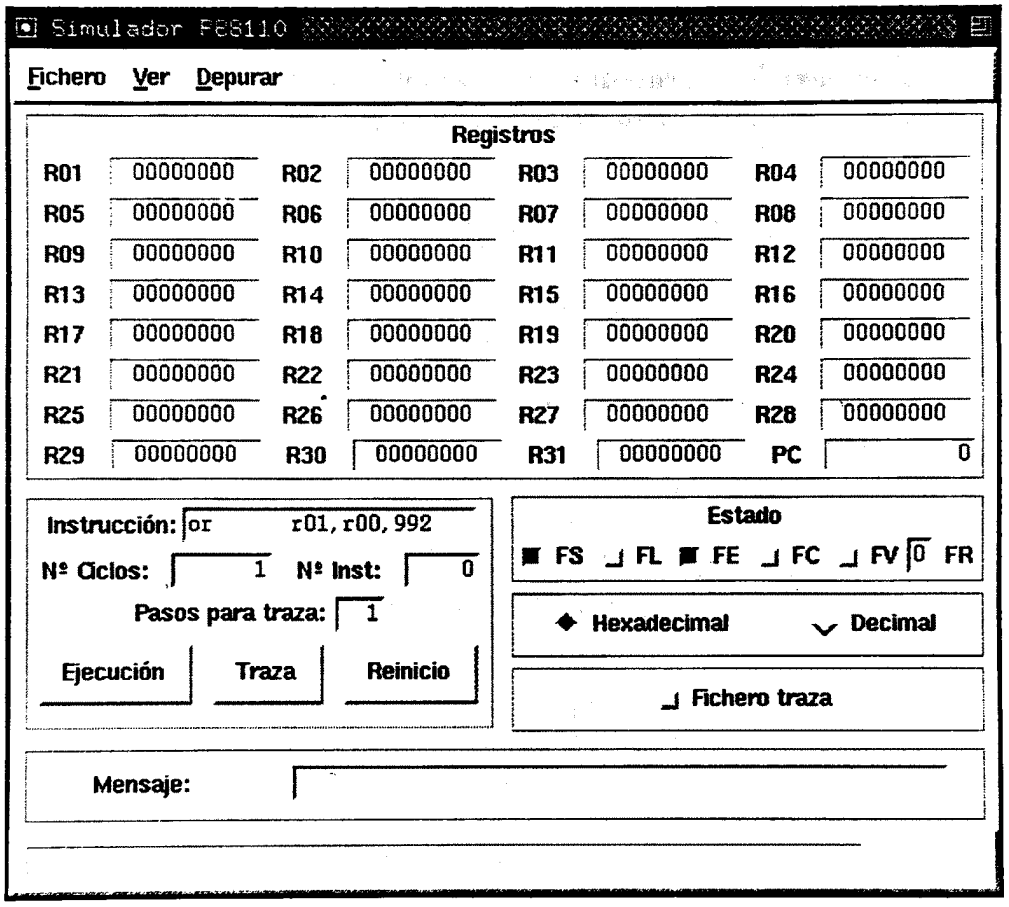

Figura 2: Inicio del emulador en la versión con X-window.

En ambos casos, el emulador dispone de un depurador que ayuda al alumno a controlar la ejecución de su programa, proporcionándole las opciones clásicas de ejecución ciclo a ciclo, establecimiento de puntos de ruptura y visualización del contenido de registros y posiciones de memoria.

3 Análisis de prestaciones de memorias cache

Esta práctica corresponde a la asignatura de Arquitectura de Computadores, que se imparte en el primer cuatrimestre de tercer curso, y cuyo contenido práctico es de aproximadamente el 33%.

Con esta práctica se pretende que el alumno analice la influencia de diversos parámetros de diseño de la memoria cache sobre las prestaciones de uno de los componentes críticos del computador como es su sistema de memoria. Parámetros tales como los tamaños de las memorias cache y de sus bloques así como las distintas políticas de ubicación, reemplazo y escritura.

Por otra parte, otro de los objetivos de esta práctica es que el alumno emplee algunas de las técnicas de transformación de programas existentes en la actualidad, cuya finalidad es la de mejorar las prestaciones del sistema de memoria. Entre ellas se aplicarán, por una parte aquellas orientadas a reducir el número de peticiones al sistema de memoria (p.e. el reemplazo escalar) y por otra, aquéllas cuyo objetivo es minimizar el número de fallos de las memorias cache. En este último caso se contemplarán los distintos tipos de fallos que se pueden producir: carga inicial, conflictos y capacidad.

En esta práctica se proporcionará al alumno un programa en ensamblador del procesador Motorola 88110 que realice una determinada operación sobre matrices, y que llevará ya incluida una optimización clásicamente realizada por cualquier compilador que genere código optimizado: la asignación a registros de las variables utilizadas como subíndices de las matrices.

En primer lugar, el alumno deberá determinar el tamaño más adecuado de la cache de datos, de modo que se minimice el número de fallos con el menor coste posible. Para ello se considerarán fijos los distintos parámetros de la cache de datos: tamaño de bloque, organización (se propondrá la directa) y política de escritura. Además se deberán determinar los fallos debidos a carga inicial y los fallos debidos a conflictos.

En el programa que se dará al alumno, se habrán asociado direcciones a las matrices de modo que no estén alineadas a bloque, para que él compruebe cómo al aumentar el tamaño de la cache no se alcanza el mínimo número de fallos, determine los fallos debidos a conflictos y capacidad, y modifique la ubicación de las estructuras para conseguirlo.

El alumno deberá configurar en este caso el emulador, activando sus memorias cache con los parámetros concretos que se le proponen e invocará al emulador para ejecutar el programa previamente ensamblado. El emulador proporciona en este caso las estadísticas obtenidas acerca de las memorias cache, a lo largo de la ejecución del programa, pudiéndose visualizar éstas bien ejecutando ciclo a ciclo o bien ejecutando de principio a fin sin paradas. La figura 3 muestra la salida del emulador una vez finalizada la ejecución de un programa.

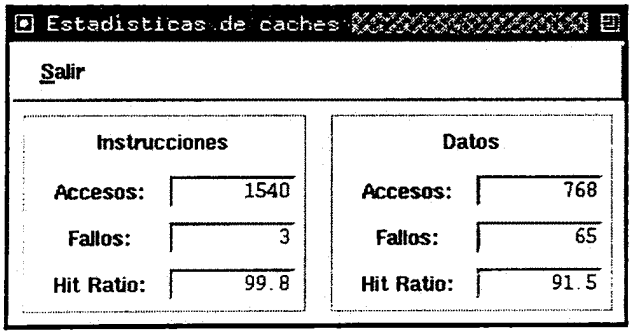

Figura 3: Estadísticas de las memorias cache en la versión con X-window.

Una vez realizada esta primera toma de contacto con el funcionamiento práctico de las memorias cache del emulador se propondrá modificar los distintos parámetros de la cache de datos, variando los parámetros 0 indicados para que el alumno pueda extraer conclusiones acerca de la configuración más adecuada para el ejemplo propuesto.

Como dijimos anteriormente, otro de los objetivos de esta práctica es que el alumno aplique algunas de las técnicas de transformación de programas, explicadas brevemente en la parte teórica de la asignatura, como por ejemplo *merging arrays, loop interchanging, loop unrolling, blocking*, etc. En este caso, puesto que el alumno conoce el juego de instrucciones del procesador emulado, será él mismo el que modificará el programa original que se le proporcionó, para aplicar la optimización concreta, y comprobará el efecto que produce, utilizando de nuevo el emulador. Para ello, configurará dichas memorias con los parámetros óptimos obtenidos mediante las pruebas anteriormente realizadas.

Hay que hacer notar las ventajas que supone la utilización de este emulador sobre simuladores de cache como dineroIII. Este último acepta un fichero de traza, previamente generado mediante el simulador dlxsim, por lo que el tamaño ocupado por este fichero hace prohibitiva su utilización para programas relativamente largos. En el caso de em88110 no se genera este fichero sino que se van acumulando las estadísticas a medida que avanza la ejecución del programa, por lo que permite evaluar programas con mayor número de referencias a memoria sin coste adicional de ocupación de disco.

4 Análisis de prestaciones en computadores segmentados y superescalares

Al igual que la anterior, esta práctica corresponde a la asignatura de Arquitectura de Computadores. Se plantea cuando el alumno ya conoce, aunque únicamente a nivel teórico, los conceptos básicos del *pipeline* de instrucciones, los fundamentos de los procesadores superescalares, y los distintos tipos de dependencias que pueden existir entre instrucciones así como los efectos negativos que estas dependencias pueden ocasionar en el rendimiento del computador.

El objetivo, en este caso, es que el alumno analice la influencia de algunas características típicas de los procesadores actuales, en el tiempo empleado en la ejecución de un programa, así como en el número medio de ciclos por instrucción resultante. Entre estas características cabe destacar: el número de etapas de *pipeline* que tiene, el número de unidades funcionales de que dispone y si éstas están o no segmentadas, así como su capacidad de emitir varias instrucciones por ciclo.

Por otra parte, se pretende que el alumno analice la mejora que se puede conseguir, utilizando instrucciones de bifurcación retardada y de predicción de salto, para pasar finalmente a aplicar técnicas de compilación como por ejemplo *loop unrolling* o *software pipelining* al programa, evaluando la mejora que se puede obtener con su utilización.

Para esta práctica se proporcionará al alumno un bucle sencillo, programado en ensamblador del 88110, sin tener en cuenta las características anteriormente mencionadas, y que funcionaría correctamente si el emulador estuviera configurado del modo que se 10 en la práctica de la asignatura de segundo curso. A partir de este programa se propondrá en primer lugar, que el alumno ejecute dicho programa en el emulador, configurado en este caso para trabajar en modo superescalar y con las memorias cache activadas. En dicha ejecución, se deberán identificar los distintos parones que se producen en el *pipeline*, debidos a las dependencias existentes en el programa (tanto de datos como estructurales), a las instrucciones de bifurcación y a las distintas latencias de las unidades funcionales, calculándose el número medio de ciclos por instrucción.

A continuación, se propondrá la utilización de instrucciones de bifurcación retardada y de predicción de salto, evaluándose en este último caso el efecto que producen los fallos en la predicción. En el emulador no se han incorporado los mecanismos de predicción dinámica de que dispone el M88110, por lo que el alumno dispondrá únicamente de las instrucciones de bifurcación retardada y predicción estática de salto.

Finalmente, el alumno deberá transformar el programa original, aplicando alguna de las técnicas de compilación antes mencionadas, y reordenará el código resultante, utilizando asimismo las instrucciones de bifurcación retardada y predicción de salto que considere convenientes, de modo que se minimice el número de parones. Se deberá tener en cuenta además, a la hora de hacer la reordenación de código, que el emulador no dispone por el momento de mecanismos de adelantamiento o *forwarding*, que sí están incluidos en el procesador original. El programa resultante, se ejecutará de nuevo sobre el emulador, calculándose el nuevo CPI e identificándose la mejora obtenida sobre la solución anterior.

Como hemos dicho, para la realización de esta práctica se deberá configurar el emulador

para trabajar en modo paralelo (superescalar), activando las memorias cache de una forma predeterminada. El emulador permitirá seguir la evolución de las O por las distintas etapas del *pipeline*, y presentará las estadísticas de las memorias cache en cada momento. Esto último es de gran importancia en esta práctica puesto que los fallos, tanto en la cache de datos como en la de instrucciones, producen a su vez parones en el *pipeline*. La figura 4 muestra una vista bastante completa de lo que el emulador ofrece al usuario para facilitar la comprensión de los conceptos que aquí se estudian:

- El listado del programa ensamblador en ejecución, junto con las direcciones que ocupa cada instrucción.
- El estado del *pipeline*. En esta figura se puede ver cómo las dos primeras instrucciones del programa entran a la vez en la fase de decodificación.

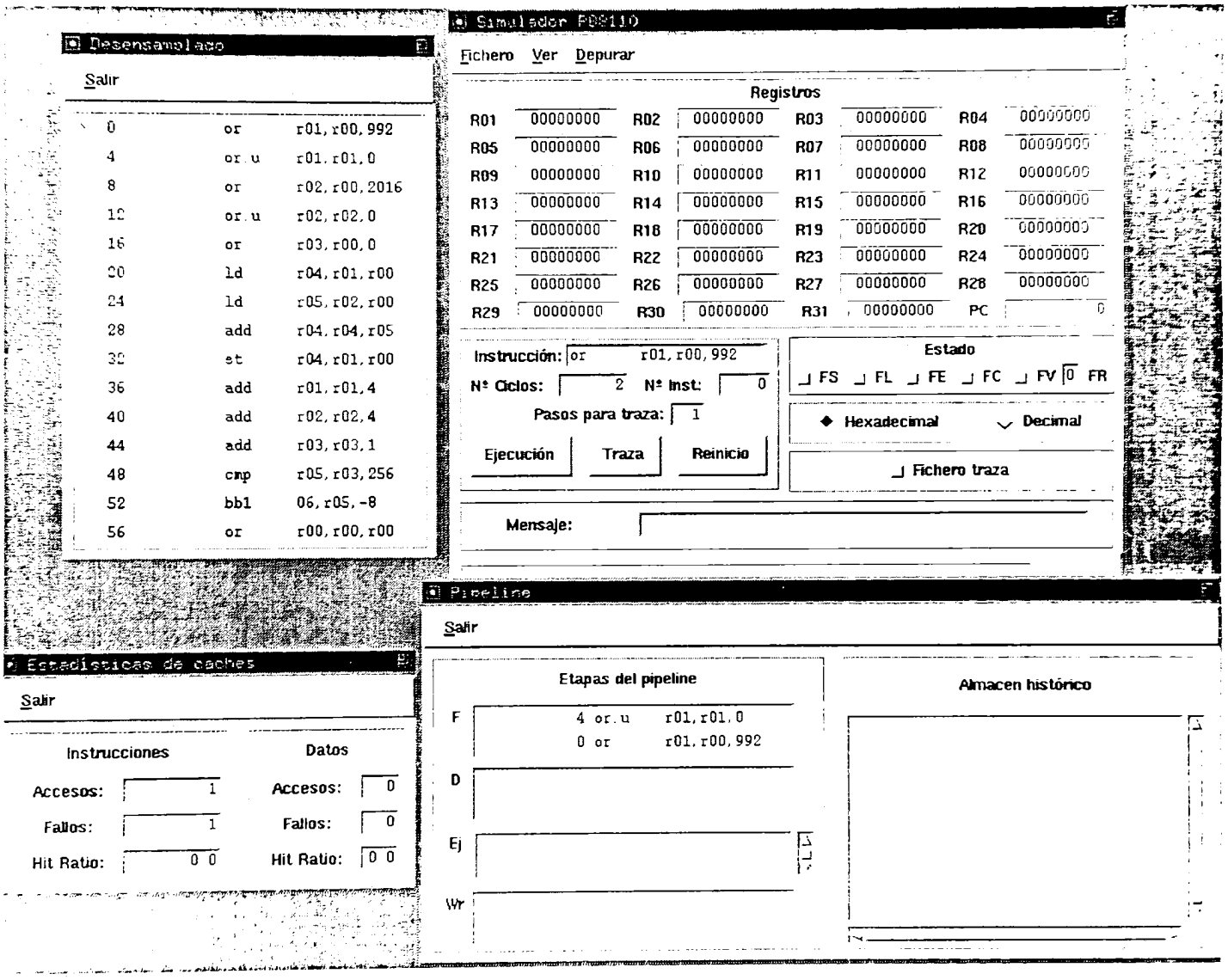

Figura 4: Utilización del emulador en modo superescalar.

- El contenido del almacén histórico. Éste es el mecanismo que utiliza el MS8110 para poder volver hacia atrás cuando hay un error en la predicción de salto. En él se almacenan las instrucciones que se han ejecutado de forma errónea, así como el contenido original de los registros que éstas han modificado.
- Las estadísticas de las memorias cache. En la figura se puede ver que se ha producido un fallo en la cache de instrucciones al realizar la búsqueda de las 2 primeras instrucciones.

5 Referencias

[1] Departamento de Arquitectura y Tecnología de Sistemas Informáticos. *Prácticas de Fundamentos de los Computadores: Programación en Ensamblador*. Marzo 1997.

[2] Keith Diefendorff. Michael Allen. Organization of the Motorola 88110 superscalar RISC microprocessor. *IEEE Micro*. 12(2):40–63. Abril 1992.

[3] John L. Hennessy. David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers. Los Altos. CA 94022, USA, 2ª edición. 1995.

[4] Mehdi Zargham. *Computer Architecture: Single and parallel systems*. Prentice Hall. 1996.

[5] Motorola. *MC88110: Second Generation RISC Microprocessor. User's Manual*. Motorola Inc.. 1991.