SIMULADOR PARA LA ENSEÑANZA DE LA PLANIFICACIÓN DINÁMICA DE INSTRUCCIONES

Sergio Sáez[1], Juan Luis Posadas[1], Pedro López[1]

[1]*Departamento de Informática de Sistemas y Computadores (DISCA)*
Facultad de Informática, Universidad Politécnica de Valencia
E-mail: { ssaez, jposadas, plopez } @disca.upv.es

RESUMEN: El presente trabajo ilustra el desarrollo y utilización de un simulador para la enseñanza de la gestión dinámica de instrucciones en un procesador segmentado a alumnos de cuarto curso de Ingeniería Informática. El objetivo de dicho simulador es facilitar la comprensión del algoritmo de Tomasulo y del hardware necesario para su implementación y funcionamiento, así como proporcionar la posibilidad de visualizar el estado del procesador (instrucción en cada etapa, bancos de registros de enteros y de coma flotante, estaciones de reserva, memoria, etc.) al final de cada uno de los ciclos de reloj necesarios para la ejecución de un determinado conjunto de instrucciones. Este simulador se utiliza en las prácticas de la asignatura de Arquitectura de Computadores II de la titulación de Ingeniero de Informática de la UPV.

1. INTRODUCCIÓN

En los actuales planes de estudios para la obtención del título de Ingeniero de Informática (II) en la Universidad Politécnica de Valencia (UPV), el estudio de la arquitectura del computador es competencia de dos asignaturas de segundo ciclo de carácter troncal, Arquitectura de Computadores 1 (ARQ1) y Arquitectura de Computadores 2 (ARQ2) [5], ubicadas en el séptimo y octavo semestre de la titulación, respectivamente.
Previamente a estas asignaturas el alumno ha cursado en el primer ciclo las materias Estructura de Computadores 1 (EC1) y Estructura de Computadores 2 (EC2) ubicadas en el segundo y tercer semestres, respectivamente, donde se estudia la organización general del computador y ya se introduce el concepto de procesador segmentado [1].
En la asignatura ARQ1 se define el concepto moderno de arquitectura, y se aplican las técnicas de segmentación a la unidad de instrucción del computador. Sólo se consideran las instrucciones enteras y los riesgos se solucionan mediante técnicas sencillas (inserción de ciclos de parada y adelantamiento).

La asignatura ARQ2 es una continuación de ARQ1 que comprende los siguientes objetivos:

- Aplicar las técnicas de segmentación a la unidad de instrucción del computador con operaciones multiciclo. Diferenciar las técnicas de **gestión** estática vs. **dinámica de**

instrucciones. Conocer y comprender las técnicas de gestión especulativa de instrucciones. Conocer y comprender el concepto de procesador superescalar.

- Distinguir los tipos de computadores orientados al procesamiento de vectores. Conocer y comprender los computadores vectoriales segmentados.
- Diferenciar los multiprocesadores de memoria compartida y los multicomputadores.
- Identificar los aspectos que más influencia tienen sobre las prestaciones de éstos.

La organización de la docencia de ambas asignaturas corresponde con 3 créditos de teoría y problemas y 1,5 créditos de trabajo en el laboratorio. Dicho trabajo consiste en la realización, de un modo coordinado con las clases de teoría y problemas, de un conjunto de prácticas que refuerzan la comprensión y aplican los conceptos estudiados.

En este trabajo se presenta un simulador desarrollado específicamente para la realización de la segunda práctica de la asignatura ARQ2 que trata sobre la planificación dinámica de instrucciones basada en el algoritmo de Tomasulo. Previamente, el alumno ya ha trabajado en la primera con la gestión estática de instrucciones. El simulador utilizado permite al alumno realizar la implementación de un planificador basado en el algoritmo de Tomasulo, así como comprobar su funcionamiento. Para ello, se proporciona al alumno el código de un simulador en el que falta por escribir el código correspondiente al planificador de instrucciones, trabajo que éste debe desarrollar en el laboratorio. Para verificar el correcto funcionamiento, la herramienta desarrollada permite la visualización ciclo a ciclo de la ejecución de las instrucciones, observándose el contenido de las diferentes etapas junto con el estado interno del procesador. La principal contribución de esta herramienta es precisamente la posibilidad de, no sólo analizar el funcionamiento de un planificador como permiten otras herramientas [2] sino también realizar su implementación de forma sencilla. El tiempo disponible para la realización de la práctica es de dos sesiones de dos horas cada una.

Este trabajo está organizado como sigue: en la sección 2 se presenta el simulador y el trabajo que debe realizar el alumno en el laboratorio. En la sección 3 se presenta el funcionamiento del simulador y los resultados y ficheros para visualización que genera. Finalmente, en la sección 4 se exponen algunas conclusiones de la experiencia.

2. SIMULADOR DE UN PROCESADOR SEGMENTADO CON PLANIFICACIÓN DINÁMICA DE INSTRUCCIONES

El objetivo del simulador presentado en este trabajo es ofrecer al alumno la posibilidad de analizar de una forma didáctica y visual el correcto funcionamiento del algoritmo de Tomasulo para la ejecución de instrucciones con operaciones multiciclo donde se permite alterar su orden de lanzamiento a ejecución. El simulador ha sido implementado en lenguaje C bajo el sistema operativo LINUX.

El algoritmo de Tomasulo descompone la ejecución de instrucciones en tres fases. En primer lugar en la fase de lanzamiento (fase Issue), se decodifica la instrucción, se envía a la estación de reserva del operador correspondiente, y se comprueban las dependencias de datos, transfiriendo desde los registros los operandos que estén disponibles (para prevenir riesgos WAR) o bien copiando el identificador (marca) del operador del que depende la instrucción en curso (para resolver riesgos RAW). También se reserva el registro destino, indicándole el

identificador del operador correspondiente. Seguidamente, en la fase de ejecución, los operadores toman instrucciones de las estaciones de reserva que tengan todos sus operandos disponibles, y realizan la operación correspondiente. En caso de que ninguna estación de reserva tenga sus operandos disponibles, el operador está ocioso. Finalmente, en la fase de escritura de resultados (fase Write Back), los operadores escriben sobre el bus común de datos el resultado de la operación y su identificador, de tal manera que todas la operaciones pendientes de ese resultado lo toman del bus y lo transfieren a la estación de reserva correspondiente.

Al alumno se le proporciona un simulador en el que el código correspondiente a las fases de lanzamiento y escritura de resultados se ha eliminado, dejando únicamente un esqueleto de dichos procedimientos. También se le proporciona un algoritmo en pseudocódigo en el que se definen todas la acciones que tienen lugar en cada una de las fases del algoritmo. El trabajo del alumno consiste en escribir dichos procedimientos y conseguir un simulador que funcione.

La práctica por tanto está dividida principalmente en dos apartados: una primera parte donde el alumno debe familiarizarse con la estructura de datos utilizada en la implementación del simulador y acabar de desarrollar el código correspondiente al algoritmo de Tomasulo, y una segunda parte donde el alumno tiene que comprobar el funcionamiento correcto del código que ha generado simulando la ejecución de varios programas en ensamblador. La primera parte de implementación obliga al alumno a comprender cómo funciona el algoritmo así como analizar qué recursos se necesitan para diseñar un procesador que lo aplique, debiendo proporcionar soluciones a problemas tales como la escritura simultánea en el bus común de datos por parte de los operadores, los cuales no se abordan con los métodos tradicionales. En la segunda parte, con la simulación de la ejecución de programas y posterior visualización de los resultados obtenidos, el alumno puede comprobar si la versión del algoritmo que ha realizado es correcta.

Los resultados que se generan en una simulación son ficheros ".html" que permiten visualizar el cronograma de ejecución de las instrucciones junto con el estado del procesador ciclo a ciclo. En el caso de obtenerse un funcionamiento incorrecto, el alumno puede observar paso a paso la evolución de la ejecución de las instrucciones y detectar dónde falla el algoritmo, para así proceder a su rectificación y volver a simular. Este método permite profundizar de forma sencilla en el estudio y comprensión del algoritmo al poderse comprobar de forma práctica y visual cualquier situación teórica.
La evaluación de la práctica y comprensión que el alumno tiene que haber adquirido se realiza a través de un cuestionario escrito con preguntas relativas a la ejecución de los programas ejemplo y así forzar a su seguimiento visual ciclo a ciclo.

La experiencia llevada a cabo ha sido satisfactoria y los alumnos agradecen el material disponible ya que les facilita la comprensión de los conceptos estudiados en clase, además de permitirles preparar el examen con mayor facilidad al disponer de múltiples ejemplos de ejecución que ellos mismos pueden generar y visualizar tantas veces como necesiten.

3. VISUALIZACIÓN DE LOS RESULTADOS OBTENIDOS

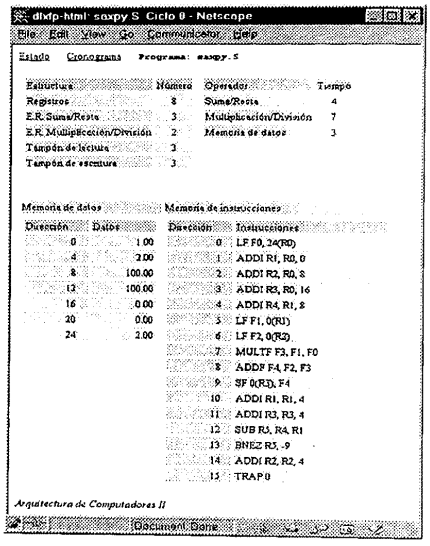
El simulador, denominado DLX/FP, está basado en el procesador DLX [3] [4] con instrucciones de coma flotante. Permite interpretar código ensamblador desarrollado con un conjunto de instrucciones enteras reducido, e instrucciones de coma flotante aritméticas y de carga/almacenamiento de simple precisión.

Una vez implementado y compilado el simulador, se comprueba su funcionamiento mediante una serie de ejemplos. Uno de ellos es el código en ensamblador correspondiente al bucle que

realiza el producto escalar de un entero "*a*" por un vector "*x*" y elemento a elemento se suma el resultado con un vector "*y*". En el código desarrollado los vectores sólo son de 2 elementos (lo que supone dos iteraciones en el bucle) pues es suficiente para comprobar el funcionamiento del algoritmo y así se reduce el número de ficheros resultado. El nombre del fichero en ensamblador a ejecutar se pasa como parámetro al simulador (por ejemplo: dlxfp –f saxpy.s). Entonces, el simulador genera un fichero "index.html" con las características del simulador y el código que se ha ejecutado y, además, dos ficheros "ciclox.html" y "estadox.html" por cada ciclo de reloj que se necesitaría en la ejecución real del código, donde x indica el número de ciclo.

De esta forma, los resultados de la simulación quedan almacenados en disco, y utilizando un visualizador html puede reproducirse de forma sencilla la ejecución del código ensamblador mediante la presentación del cronograma de las instrucciones y del estado del procesador.

El fichero "index.html" para el ejemplo indicado puede observarse en la figura 1. Presenta la configuración con la que se ha realizado la simulación: un banco de ocho registros de coma flotante, tres estaciones de reserva de suma y resta, dos estaciones de reserva de multiplicación y división, tres tampones de lectura, tres tampones de escritura, un operador de suma y resta con un tiempo de cuatro ciclos, un operador de multiplicación y división con un tiempo de siete ciclos y, por último, una memoria de datos con un tiempo de acceso de tres ciclos donde puede apreciarse su contenido una vez inicializada. Además, también se presenta la memoria de instrucciones con el código ensamblador simulado. En este caso puede observarse el código correspondiente al bucle *saxpy*. Es importante destacar que el simulador permite cambiar la configuración para así poder comparar diferentes resultados dependientes de ésta (número estaciones de reserva, tiempo de los operadores, etc.).


dlxfp-html: saxpy S Ciclo 0 - Netscape

File Edit View Go Communicator Help

Estado Cronograma Programa: saxpy.S

Estructura	Número	Operador	Tiempo
Registros	8	Suma/Resta	4
E.R. Suma/Resta	3	Multiplicación/División	7
E.R. Multiplicación/División	2	Memoria de datos	3
Tampón de lectura	3		
Tampón de escritura	3		

Memoria de datos

Dirección	Datos
0	1.00
4	2.00
8	100.00
12	100.00
16	0.00
20	0.00
24	2.00

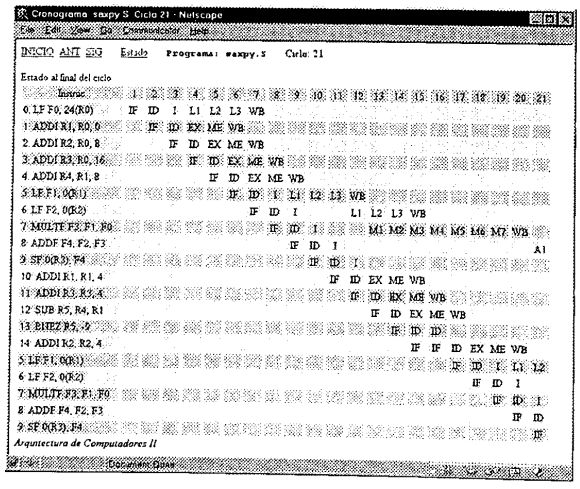
Memoria de instrucciones

Dirección	Instrucciones
0	LF F0, 24(R0)
1	ADDI R1, R0, 0
2	ADDI R2, R0, 8
3	ADDI R3, R0, 16
4	ADDI R4, R1, 8
5	LF F1, 0(R1)
6	LF F2, 0(R2)
7	MULTF F3, F1, F0
8	ADDF F4, F2, F3
9	SF 0(R3), F4
10	ADDI R1, R1, 4
11	ADDI R3, R3, 4
12	SUB R5, R4, R1
13	BNEZ R5, -9
14	ADDI R2, R2, 4
15	TRAP 0

Arquitectura de Computadores II

Document Done

Figura 1: fichero index.html


Cronograma saxpy S Ciclo 21 - Netscape

File Edit View Go Communicator Help

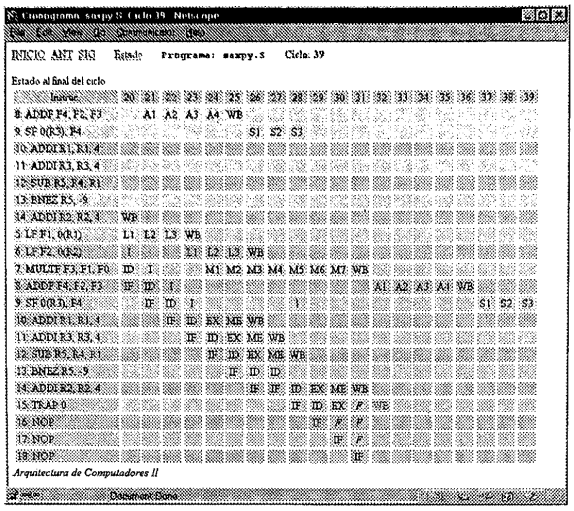
INICIO ANT SIG Estado Programa: saxpy.S Ciclo: 21

Estado al final del ciclo

Instruc	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0 LF F0, 24(R0)	IF	ID	I	L1	L2	L3	WB														
1 ADDI R1, R0, 0		IF	ID	EX	ME	WB															
2 ADDI R2, R0, 8			IF	ID	EX	ME	WB														
3 ADDI R3, R0, 16				IF	ID	EX	ME	WB													
4 ADDI R4, R1, 8					IF	ID	EX	ME	WB												
5 LF F1, 0(R1)						IF	ID	I	L1	L2	L3	WB									
6 LF F2, 0(R2)							IF	ID	I			L1	L2	L3	WB						
7 MULTF F3, F1, F0								IF	ID	I					M1	M2	M3	M4	M5	M6	M7
8 ADDF F4, F2, F3									IF	ID	I										A1
9 SF 0(R3), F4										IF	ID	I									
10 ADDI R1, R1, 4											IF	ID	EX	ME	WB						
11 ADDI R3, R3, 4												IF	ID	EX	ME	WB					
12 SUB R5, R4, R1													IF	ID	EX	ME	WB				
13 BNEZ R5, -9														IF	ID	ID					
14 ADDI R2, R2, 4															IF	IF	ID	EX	ME	WB	
5 LF F1, 0(R1)																	IF	ID	I	L1	L2
6 LF F2, 0(R2)																		IF	ID	I	
7 MULTF F3, F1, F0																			IF	ID	I
8 ADDF F4, F2, F3																				IF	ID
9 SF 0(R3), F4																					IF

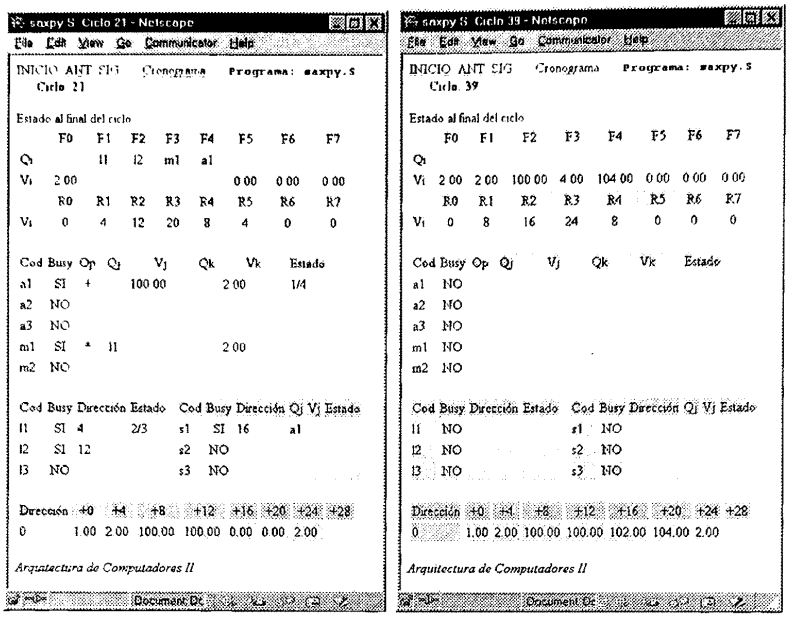
Arquitectura de Computadores II

Document Done



Figuras 2 y 3: ficheros ciclo21.html y ciclo39.html

En las figuras 2 y 3 puede observarse el cronograma de ejecución de las instrucciones correspondiente al bucle *saxpy*. El simulador ha generado un total de 39 ciclos de reloj que son, empleando la configuración indicada, el tiempo global de ejecución del código ejemplo. En las figuras se muestra el estado del cronograma en el ciclo 21 y en el último ciclo 39. Puede observarse que cada ventana dispone en la parte superior de un enlace al ciclo anterior y otro al posterior, permitiendo fácilmente avanzar o retroceder ciclo a ciclo en la ejecución de las instrucciones. Viendo los diferentes cronogramas generados (uno por cada ciclo) se comprueba cómo el *hardware* lanza a ejecución instrucciones posteriores a otras que se han parado y cómo se altera dinámicamente el orden de la ejecución, evitando que el hecho de parar una instrucción afecte a las que le siguen. En los cronogramas también puede apreciarse la existencia de otro enlace con nombre *estado* que sirve para abrir una nueva ventana donde se visualiza el estado del procesador en ese ciclo.



Figuras 4 y 5: estado del procesador en ciclos 21 y 39

En las figuras 4 y 5 pueden verse las ventanas correspondientes al estado del procesador al final de los ciclos de reloj 21 y 39.
En las ventanas del estado del procesador se presenta: el banco de registros de enteros; el banco de registros de coma flotante con las marcas (Qj, Qk) en ese instante asociadas a cada uno de ellos (la marca de un registro depende de la estación reservada por la instrucción que va a escribir en dicho registro); las estaciones de reserva, indicándose las operaciones que las ocupan, la disponibilidad de los operandos (valores, o, en su defecto, las marcas correspondientes) y el número de ciclos ejecutados en el caso de estar en ejecución dichas operaciones; los tampones de lectura y escritura con las direcciones de acceso; y, por último, el contenido de la memoria de datos para permitir comprobar si al final de la ejecución del código se han obtenido los resultados correctos.

4. CONCLUSIONES

En este trabajo se ha presentado un simulador para la gestión dinámica de instrucciones que se está utilizando en las prácticas de la asignatura Arquitectura de Computadores 2 de la Facultad de Informática de la UPV. El uso de dicho simulador ha facilitado enormemente al alumno el estudio de los algoritmos para la planificación dinámica de instrucciones al permitirle modificar fácilmente su código para implementar un algoritmo de planificación así como visualizar ciclo a ciclo el cronograma de ejecución de las instrucciones y observar al mismo tiempo el estado del procesador al final de cada uno de los ciclos de reloj.

El alumno puede comprobar de una forma práctica y con diferentes configuraciones del procesador como se altera dinámicamente el orden de ejecución de las instrucciones

permitiéndose que instrucciones posteriores a una parada no se detengan y acaben antes su ejecución.

REFERENCIAS

[1] J.L. Posadas, A. Robles. Innovación Educativa en la Enseñanza del Procesador Segmentado. JENUI 99

[2] J.L. Hennessy, D.A. Patterson. Herramientas y Simuladores: http://www.mkp.com/books_catalog/ca/hp2e_res.asp#other

[3] J.L. Hennessy, D.A. Patterson. Arquitectura de Computadores, un enfoque cuantitativo. New York. Mc. Graw Hill. 1993

[4] D.A. Patterson, J.L. Hennessy. Organización y Diseño de Comput. Mc. Graw Hill. 1995

[5] WEB de las asignaturas ARQ1 y ARQ2: http://dlxv.disca.upv.es