Simulación acotada de procesadores superescalares orientados a la ensañanza

Tomás Juárez[1], Guillermo Pacheco[1], Ing. Martín Menchón[1], and Ing. Marcelo Tosini[1]

[1] *Universidad Nacional del Centro de la Provincia de Buenos Aires*

29 de agosto de 2016

Resumo

Los cursos de arquitectura de computadores actuales tienen como objetivo principal brindar al estudiante las bases del funcionamiento del hardware que subyace en los dispositivos modernos. Uno de los inconvenientes que surgen al abordar esta temática involucran conceptos pedagógicos, tales como la dificultad de realizar un seguimiento del comportamiento del ordenador con el fin de comprender su funcionamiento. En particular, este trabajo se basó en la representación esquemática de la ejecución de un procesador superescalar básico haciendo énfasis en el caso de emisión y ejecución en desorden. Esto resulta complejo de representar para el alumno en los ejercicios prácticos, con lo cual se implementó un sistema acotado con el fin de simular ciertos aspectos de un procesador superescalar genérico con un funcionamiento sencillo con el fin de subsanar este inconveniente.

Palabras-clave: superescalar. buffer de reordenamiento.

Introducción

Los procesadores actuales suelen ser muy complejos para la enseñanza en los cursos de arquitectura de computadoras, motivo por el cual se introducen sistemas pequeños que contengan la esencia de las arquitecturas modernas pero acotando su funcionalidad para enfocar al estudiante en los elementos clave.

En el caso particular de los procesadores superescalares, resulta dificu ltoso poder entender su labor interna dado que involucran conceptos avanzados de técnicas digitales, recordando conceptos base tales como la segmentación, el funcionamiento y los riesgos del procesador escalar, jerarquía de

memorias, entre otros. Asimismo, se deben reconocer nuevos riesgos que en los procesadores escalares con emisión y ejecución en orden no afectaban a su funcionamiento, problemas por fallos en memorias (puesto que el procesador superescalar agrega entre algunas de sus etapas pequeñas memorias caché para aumentar su rendimiento), algoritmos de planificación, especulación de saltos, tratamiento de excepciones, entre otros.

Es menester que el estudiante comprenda que el objetivo del curso es poder acercarse al diseño de sistemas de hardware o como mínimo poder utilizar el conocimiento de estos para enriquecer el desarrollo de software con el fin de aumentar la eficiencia aprovechando los recursos disponibles.

Una de las partes centrales en un procesador superescalar es el llamado búfer de reordenamiento o ROB, por sus siglas en inglés. Esta pequeña memoria circular es imprescindible cuando el procesador emite y ejcuta instrucciones fuera de orden, siendo además un elemento necesario para algoritmos de planificación como el algoritmo de Tomasulo.

En el presente se evaluará un método para representar un procesador superescalar sencillo, el cual dispone de un conjunto de unidades funcionales especializadas, una unidad de dispatcher y ciertas estaciones de reservas, junto con el búfer de reordenamiento en un programa que le permitirá al estudiante resolver ejercicios en forma de guía y eventualmente corregir sus prácticas.

1 Marco Teórico

1.1 Procesadores Superescalares Fuera de Orden

Previamente se ha referenciado muchas veces a este tipo de procesadores. Se llama procesador superescalar a un procesador con planificación dinámica de la ejecución múltiple de instrucciones. En los diseños más simples de los superescalares, las instrucciones se ejecutan en orden. De todas formas, para alcanzar buenas prestaciones, se requiere que el compilador modifique el código a ejecutar (tanto como sea posible), para eliminar algunas dependencias, entre otras tareas. Incluso habiendo optimizado el código por medio del compilador (planificación estática), un procesador superescalar asegura la correcta ejecución del código de entrada por medio de hardware especializado, dando la ventaja de que el código funcionará de forma correcta independientemente de la estructura del pipeline o de la capacidad del procesador. Esta es una de las tantas diferencias entre un superescalar y un procesador VLIW, en donde este último, en algunos diseños requiere que se recompile el código cuando éste se quiere ejecutar en diferentes modelos del procesador.

Todos los procesadores superescalares modernos hacen uso de la planificación dinámica del pipeline, en donde se eligen las instrucciones a ejecutar en cada ciclo de reloj intentando en lo posible evitar los riesgos y bloqueos, básicamente reordenando el código permitiendo aprovechar los recursos del sistema.

	Instruction	Issuing	Dispatch	Write
S1	DIV F10,F0,F6	1	2	42
S2	LD F2, 40(R3)	2	3	13
S3	MUL F0,F2,F4	3	14	19
S4	SUB F8,F2,F6	4	14	15

Tabela 1 – Ejemplo de instrucciones en emisión y ejecución fuera de orden.

Los procesadores superescalares con emisión y ejecución en orden recorren el camino de datos siguiendo el orden estricto del programa y todas las dependencias de datos son resueltas antes que las instrucciones pasen a la última etapa. En un superesclar fuera de orden, las isntrucciones pueden viajar por el camino de datos y ser ejecutadas antes que otras instrucciones que las preceden en el orden del programa. Sin embargo, la elección de las instrucciones a ejecutar no es arbitraria, sino que debe tener en cuenta que existen aún riesgos por dependencia de datos verdaderas (RAW) o bien riesgos estructurales, y también de control. Para llevar a cabo este dinamismo, el pipeline ahora se divide en tres partes fundamentales: una unidad encargada de buscar y decidir qué instrucciones ejecutar en cada ciclo de reloj (issue unit), muchas unidades funcionales, y una unidad de confirmación. Cada una de las estaciones de reserva dispone de búferes llamados estaciones de reserva que almacenan la operación y los operandos. Una vez que la instrucción en cuestión se encuentre en uno de estos búferes y la unidad funcional correspondiente no esté ocupada, se podrá comenzar la ejecución. Una vez obtenido el resultado, se envía a todas las estaciones de reserva que puedan necesitarlo para poder completar sus operandos y comenzar sus ejecuciones. Asimismo, se envían a una unidad de confirmación en orden, también llamado búfer de reordenamiento.

1.2 ReOrder Buffer

Cualquier tipo de instrucción que provoque la salida del cause convencional del procesador, acarrea importantes problemas en un procesador superescalar fuera de orden; en otras palabras, los riesgos de control y las excepciones en el procesador requieren unidades para brindar una tratamiento especial a las instrucciones que las ocasionan y sus subsiguientes. Consideremos las siguientes instrucciones a ejecutar [tabla 1], en donde la columna issuing describe el ciclo en el cual se emite cada instrucción, la columna dispatch establece en qué ciclo se despacha cada instrucción y write el ciclo de finalización.

Supongamos entonces que el registro F6 tiene el valor cero; lo que se ejecutaría entonces en la instrucción S1 es una división por cero, con lo cual provoca una excepción. Supongamos además, que el procesador da cuenta de esa división en el ciclo 40. Lo habitual en estos casos es que la dirección de la instrucción S1 (PC) se guarde en un registro especial, saltar a un gestor de

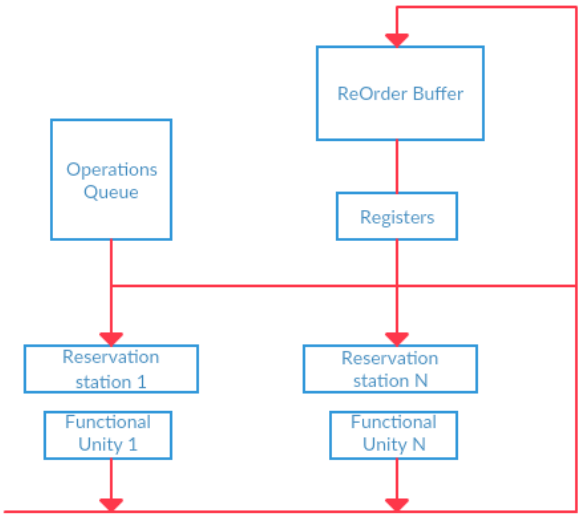

Figura 1 – Esquema del búfer de reordenamiento en un procesador superescalar.

excepciones mediante una dirección fija para estos casos, y una vez solucionado el inconveniente, se debería volver a la instrucción S1 y seguir con la ejecución desde ahí, sin embargo, esto no ocurre. Lo que realmente pasa es que en el ciclo de reloj número 40, ya hemos finalizado la ejecución de S2, S3 y S4 habiendo escrito sus resultados. La única instrucción que no se ha concretado es S1, con lo cual el registro F10 no se ha actualizado, con lo cual, al volver del tratamiento de la excepción, la instrucción S1 tendrá un nuevo valor en F0, producido por la instrucción S3. Como se puede observar, ya no hay forma de recuperar esa información con lo cual necesitamos añadir componentes a nuestro procesador para tratar esta inconsistencia. Es en este punto donde se ve la verdadera utilidad del búfer de reordenamiento [Figura 2].

El búfer de reordenamiento es una tabla de entradas con tres campos: el primero establece el registro en cuestión, una especia de indexación. El segundo campo contiene el valor producido por una instrucción determinada sobre ese registro destino, y un campo de un bit para determinar si el resultado es válido o no. Esta pequeña memoria mantiene las instrucciones en el mismo orden del programa fuente, y tiene una estructura de lista circular en la cual contiene un puntero a cabeza y cola de la misma [Figura **??**]. El puntero a cola indica la próxima entrada libre y el puntero a cabeza indica la instrucción confirmada que dejará el ROB primero. Cuando una instrucción es emitida, se inserta una entrada en secuencia en el ROB. Cada entrada contiene el estado de la instrucción; emitida (i), en ejecución (X), finished (f) además de otros elementos. Una instrucción se retira sí y sólo sí ha terminado y todas las instrucciones previas fueron retiradas.

Una vez implementado el ROB, sólo las instrucciones completadas pueden escribir en el banco de registros. En el ejemplo anterior, como S1 no

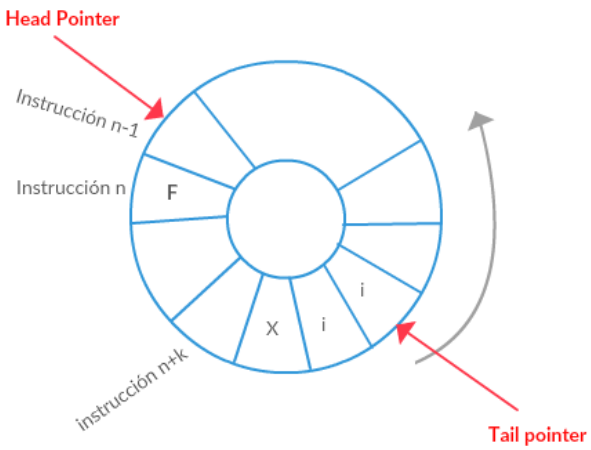

Figura 2 – Representación del búfer de reordenamiento con estructura de lista circular.

está completada, el resto de las instrucciones no podrían escribir en el banco de registros y por lo tanto, al volver a la ejecución de S1 luego del tratamiento de la excepción de división por cero, se pueden recuperar los valores de los registros, y no ocurre el problema descripto previamente.

Cabe destacar que este es el componente clave en la implementación del algoritmo de Tomasulo especulativo, el cual no es el tema de este proyecto, pero es importante remarcarlo puesto que se ve la importancia de esta memoria en las arquitecturas modernas.

2 Discusión de la Implementación

Para la implementación de la herramienta se utilizó javascript, para que el estudiante pueda utilizar la aplicación desde cualquier navegador web. La interfaz debía ser sencilla dado que el enfoque debe estar en la práctica y no requerir esfuerzos adicionales por entender la aplicación en sí misma.

Se utilizaron frameworks para mejorar la experiencia de usuario y la usabilidad de la aplicación. Lo interesante de estas bibliotecas es que uno no sólo pude utilizar funcionas ya realizadas y bien testeadas, sino que también se hereda la arquitectura. Los frameworks utilizados fueron jQuery, Bootstrap, require.js, sigma.js y ace.js, para propósito general, diseño responsivo, gestión de clases implementadas en diferentes archivos, para dibujar el grafo de dependencias y habilitar un editor de texto al usuario, respectivamente.

2.1 Parser

Cuando el usuario introduce las instrucciones, estas se deben decodificar para obtener cada una de las partes que interesan en el procesador. Como la ejecución es una simulación, lo más importante fue reconocer el tipo de instrucción y además los registros utilizados para reconocer dependencias

de datos RAW. Para lograr este objetivo se utilizó un generados de parser llamado Jison. Este genera no sólo el analizador sintáctico sino también el analizador léxico.

En este caso hay que tener en cuenta que las instrucciones a reconocer son las operaciones aritméticas de punto flotante o enteros, o bien operaciones de acceso a memoria de punto flotante o enteros. Para el caso de las operaciones aritméticas, se reconoce la suma, resta, división y multiplicación (add, sub, div, mul, addf, subf, divf, mulf), diferenciándose según el tipo de dato a operar. En el caso del acceso a memoria, estas tendrán un offset, un registro base, y un registro que contendrá el valor a leer o modificar (lw, sw, lf, sf).

3 Conclusión

En el presente trabajo se planteó el problema de la representación en los ejercicios prácticos de una procesador superescalar en papel. Esto suele ser complicado y tedioso para el alumno, con lo cual una posible ayuda es disponer de una herramienta que grafique este tipo de problemas mediante un pequelo bloque de código para ser utilizado como guía a medida que se pretenda entender el funcionamiento de un procesador moderno.

Este proyecto será utilizado como base para poder contener una mayor cantidad de funciones simuladas de un procesador genérico, es por ello que se intentó codificar de forma simple y ordenada.

4 Bibliografía

David A. Patterson, John L. Hennessy, *Estructura y diseño de computadores, la interfaz hardware/software*, Reverté, 4ta edición, 2010.

Marcelo Tosini, *Filminas de cátedra de arquitectura de computadoras y técnicas digitales*, UNCPB, 2016