Evolución de los algoritmos de ejecución fuera de orden.

Marcela Cruz, Jiangjie Chen, Beltrán Malbrán.
Facultad de Ingeniería de la Universidad de Buenos Aires.

Abstract — El presente informe se enfoca en el estudio de las ejecuciones fuera de orden, profundizando en las ventajas y desventajas de la planificación dinámica, el algoritmo del marcador y por último del algoritmo de Tomasulo. Se realiza un análisis de cómo la ejecución fuera de orden modificó la industria de los procesadores al explorar cómo estas técnicas mejoran el rendimiento y eficiencia.

Índex Terms — Ejecución fuera de orden, algoritmo de Tomasulo, algoritmo del marcador, planificación dinámica, rendimiento y eficiencia del procesador, dependencia de datos, renombre de registros.

1. INTRODUCTION.

La constante búsqueda de mejorar la eficiencia de los procesadores ha generado que se desarrollen varias técnicas tal como la ejecución fuera de orden, la cual es utilizada para mejorar el rendimiento de las instrucciones en los procesadores. Esto lo logra a través de los algoritmos planificación dinámica el cual predice el flujo del programa para determinar qué instrucción realizar teniendo en cuenta las dependencias de datos.

La ejecución fuera de orden es una posible solución a esto dado que permite que las instrucciones se ejecuten tan pronto como estén disponibles los datos.

Los eventuales problemas asociados a la ejecución fuera de orden pueden ser abordados eficazmente a través del algoritmo de Tomasulo, ingeniado por Robert Tomasulo. Este utiliza el renombrado de registros, una técnica que consiste en asignar un nombre temporal para cada registro mientras se utiliza, lo que permite mejorar la ejecución paralela de instrucciones.

En este informe analizaremos como la ejecución fuera de orden ha ido evolucionando con el pasar del tiempo, haremos énfasis en las técnicas de planificación dinámica, como están conformadas y los beneficios que estas conllevan. También detallaremos el algoritmo del marcador y como gracias a este surge el algoritmo de tomasulo, el cual ha contribuido de manera significativa a la mejora del rendimiento de los procesadores, particularmente explicaremos tanto los conceptos como etapas dentro de los algoritmos describiendo sus ventajas y desventajas. Por último veremos las aplicaciones que se le dan hoy en día al algoritmo a través de ejemplos.

Marcela Cruz - Estudiante, Licenciatura en Análisis de Sistemas - Facultad de Ingeniería UBA - mjcruz@fi.uba.ar
Jiangjie Chen -Estudiante, *Ingeniería en Informática* - Facultad de Ingeniería UBA - jchen@fi.uba.ar
Beltrán Malbrán - Estudiante, Licenciatura en Análisis de Sistemas - Facultad de Ingeniería UBA - bmalbran@fi.uba.ar

2. PLANIFICACIÓN DINÁMICA.

Para entender porque, desde su aparición, la ejecución fuera de orden se ha convertido en algo indispensable en el mundo de los procesadores primero hay que comprender la importancia de la planificación dinámica. La cual plantea un nuevo pipeline de procesamiento de instrucciones, en el cual se segmenta la etapa de decodificación de las instrucciones y comprobación de riesgos, también conocida como decode, en dos partes emisión y lectura de operandos [1].

La etapa de emisión consiste en que dada una instrucción se decodifica y comprueba los riesgos estructurales, es decir, verifica la disponibilidad de los recursos del procesador que requiere dicha instrucción. Una vez evaluados los riesgos estructurales se pasa a la etapa de lectura de operandos, en este proceso se lee de memoria siempre y cuando no haya riesgos de datos para luego ejecutarse.

Esta modificación genera un incremento en la eficiencia del procesador al permitir que las instrucciones que no poseen riesgos estructurales se procesan primero mientras que las que sí, se quedan en la etapa de emisión hasta que se resuelvan. Como consecuencia, en algunos casos, las instrucciones se ejecutan fuera de orden, lo cual permite el aprovechamiento de los retrasos imprevistos, como los debidos a los fallos de caché, ejecutando otras instrucciones mientras espera que el dato sea traído de la memoria principal tal como se ilustra en la figura 1.

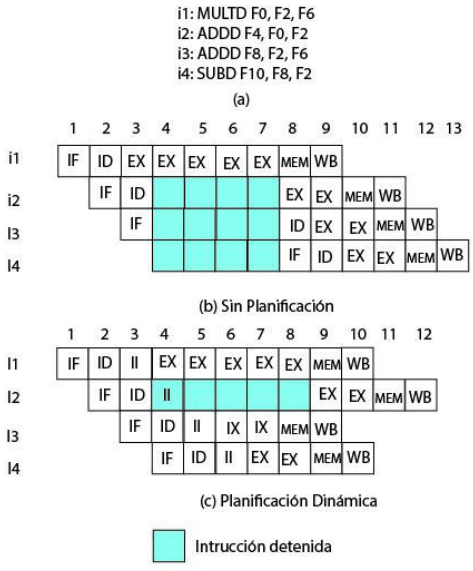

Fig. 1. Ejemplo de la ejecución de un fragmento de código sin y con planificación dinámica.

Sin embargo presenta varias desventajas siendo las más notorias la aparición de riesgos de datos, como el write after read (WAR) o el write after write (WAW), y que trae consigo una complejización notoria a la hora de tratar interrupciones [2].

Para manejar estas desventajas se han desarrollado varias técnicas, una de ellas es el método del marcador, también conocido como scoreboard, el cual es el predecesor del algoritmo de Tomasulo. Estos se distinguen en el renombre de registros y en la utilización de un bus de datos común lo cual permite la ejecución paralela de instrucciones en situaciones en las que el scoreboarding fallaría y provocaría que se detenga la ejecución.

Habiendo comprendido el funcionamiento y principalmente la importancia de la planificación dinámica profundizaremos en el algoritmo del marcador, haciendo enfoque en cómo resuelve los riesgos de WAR y WAW. Luego veremos cómo el algoritmo de Tomasulo, partiendo de las bases del scoreboard, lidia de forma más eficiente con este tipo de riesgos.

3. ALGORITMO DEL MARCADOR.

El método del marcador, también llamado Scoreboard, ejecuta las instrucciones lo antes posible siempre y cuando los registros no dependan de instrucciones activas o detenidas por riesgos estructurales en ese momento, para lograr esto utiliza un pipeline de cuatro etapas, emisión, lectura de datos, ejecución y escritura de resultados.

Para poder ejecutar las instrucciones fuera de orden teniendo en cuenta los riesgos y las dependencias se hace uso de un sistema de tablas, llamado marcador, que posee tres tablas de estado. La primera se encarga de registrar la etapa en que se encuentra cada instrucción, la segunda indica el estado de las unidades funcionales, para lograr esto mantiene constantemente actualizado nueve campos. Los cuales consisten en si la tabla está siendo utilizada, la operación a realizar, el registro de destino, el número de registro fuente, las unidades funcionales que producirán el resultado de los registros fuentes y por último informa si dichos registros están listos para su utilización. La última tabla informa que unidad funcional escribirá el resultado en cada uno de los registros. Este sistema se encarga de controlar la emisión, ejecución y detección de riesgos de las instrucciones.

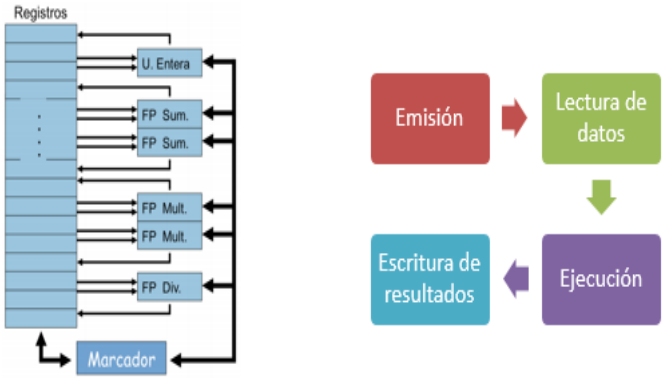

Fig. 2. Esquema de la estructura y las etapas del algoritmo del Marcador.

Si en la etapa de emisión, encargada de decodificar las instrucciones y comprobar riesgos estructurales, se ve presente un riesgo de tipo WAW el marcador tiene la responsabilidad de insertar una burbuja, esto es detener la emisión hasta que el riesgo de que dos instrucciones intenten escribir en el mismo registro en tiempo de ejecución desaparezca, en caso contrario actualiza su estructura interna y permite el paso a la etapa de lectura de datos, en la cual se verifica que no haya ninguna instrucción en actividad que vaya escribir en el operando de entrada. Esto permite evitar los riesgos de tipo read after write, también conocido como RAW por sus siglas en inglés, ya que si alguno de nuestros registros fuentes van a ser modificados no son considerados como disponibles hasta que ocurra dicha modificación. Luego de realizar esta verificación, la cual genera la posibilidad que las instrucciones entren a la etapa de ejecución de forma no ordenada, el marcador permite que la unidad funcional lea los operandos de los registros para ejecutarlos y obtener el resultado. En esta etapa es necesario tener varias unidades funcionales, las cuales pueden ser tanto simples como segmentadas, para lograr ejecutar de forma simultánea las instrucciones.

Una vez realizados todos estos procedimientos inicia la última etapa en la cual el marcador comprueba que no existan riesgos de tipo WAR, en caso de haberlos es su deber frenar la escritura del resultado hasta que todas las instrucciones que deben leer los operandos asignados a esa unidad funcional se completen. Una vez evaluados, y en caso de ser necesario solucionados, todos los riesgos se escribe el resultado en el registro destino.

A pesar de tener numerosas ventajas este algoritmo plantea un nuevo problema relacionado al espacio, ya que el necesario para la implementación del marcador puede ser tan amplio como el resto del procesador lo que genera que todos los beneficios obtenidos al lograr una ejecución fuera de orden no se vean reflejados en su plenitud, ya que si el coste es demasiado alto puede, incluso, conseguir el mismo rendimiento que la planificación estática. A pesar de esta desventaja el algoritmo del marcador se implementó en lo que se considera la primera supercomputadora exitosa, la CDC 6600,[3].

4. ALGORITMO DE TOMASULO.

Con esta desventaja en mente Robert Tomasulo diseñó en 1967 el algoritmo de Tomasulo. El cual, desde su presentación en la unidad de punto flotante en los procesadores IBM 360/91[4], es ampliamente utilizado. Este algoritmo elimina la desventaja del sistema de tablas al no utilizar dicha estructura, esto lo logró al hacer que las responsabilidades que recaen en el marcador estén distribuidas entre sus distintos componentes. Lo cual es posible gracias a la introducción de dos modificaciones significativas, el renombre de registros y el bus común de datos. A continuación explicaremos las etapas del algoritmo de Tomasulo y cómo, a diferencia de su antecesor, no detiene la ejecución frente a riesgos de tipo WAW o WAR.

Esta técnica, la cual aplica la ejecución fuera de orden gracias a su uso de planificación dinámica, elimina la etapa de lectura de datos, por lo que queda conformada por tres bloques, emisión, ejecución y escritura de resultados. A simple

vista esta división de etapas es similar a la de su antecesor sin embargo presenta, en cada una de ellas, ciertas modificaciones para que las responsabilidades que recaen en el marcador sean tratadas, de forma más eficiente, por los mismos componentes.

Para hacer uso de este algoritmo es necesario que cada unidad funcional posea una entrada, la cual es llamada estaciones de reserva, o RS por sus siglas en inglés, las cuales presentan, como máximo, cuatro campos; un campo para la operación, otro para la etiqueta de los operandos, el tercero informa el valor de estos y el último indica que estacion esta en uso.

La presencia de las estaciones de reserva conlleva a la existencia de buffers de memoria los cuales almacenan instrucciones de carga hasta que se dispone de todos los operandos y el bus de memoria está disponible para su uso. Este cambio en los componentes permite que las tareas que antes eran responsabilidad del marcador recaigan en las dos grandes modificaciones que plantea el algoritmo ideado por Tomasulo. El bus común de datos, también llamado CDB por sus siglas en inglés, y en el uso de renombre de registros en tiempo de ejecución. El bus se encarga de conectar las salidas de las unidades funcionales y la carga a memoria con el fichero de registros, las estaciones de reserva y los buffers. Es importante mencionar que en la actualidad todo bus común de datos presenta mínimo dos buses, uno para el resultado de las operaciones y otro relacionado a las etiquetas de este. La implementación del CDB genera dos consecuencias notorias, la primera es que las unidades funcionales pueden acceder al resultado de cualquier operación que tenga relación con el registro, la segunda es que las estaciones de reserva controlan cuando una instrucción puede ser ejecutada, en lugar de haber una única unidad de Detección de errores dedicada, teniendo en cuenta todo esto la estructura del hardware quedaría distribuida según la figura 3.

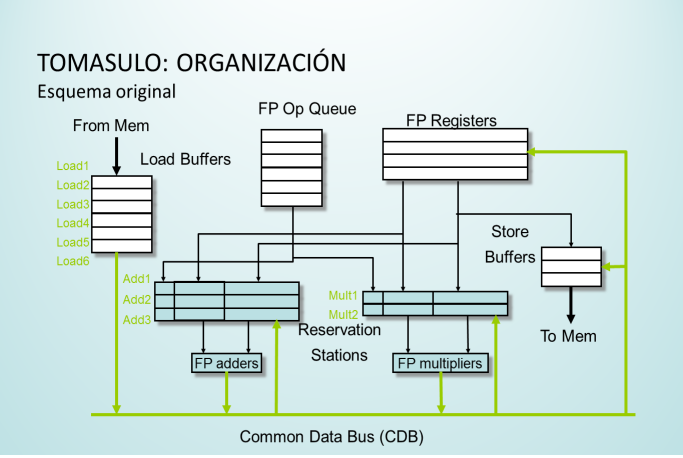

Fig. 3. Esquema de los componentes necesarios para el algoritmo tomasulo.

Después del bus de datos el siguiente cambio significativo que necesita el algoritmo de Tomasulo es el renombre de registros, esto es utilizado para ejecutar de forma correcta, gracias a la implementación de las etiquetas en las estaciones de reserva, las instrucciones que no tienen orden. Esta técnica entra en funcionamiento cuando en la etapa de emisión no se encuentra disponible un valor, generando un riesgo de tipo WAW o WAR, en estos casos se utiliza una etiqueta indicando que estación de reserva es la encargada de obtener el valor necesario. Una vez que la unidad funcional termina de transmitir el resultado en el CDB la etiqueta es reemplazada por el valor real, esto es un aumento en la eficiencia ya que no es necesario la detención de la etapa frente a riesgos de tipo WAW y WAR.

Una vez explicados los cambios presentes que conlleva la aplicación del algoritmo de Tomasulo, procederemos a explicar qué modificaciones ocurren en cada etapa.

1. En la etapa de emisión en caso de haber presente algún riesgo, se utiliza el renombre de registros y si no está disponible la estación de reserva correspondiente al tipo de operación a realizar se detiene la instrucción.

2. En la etapa de ejecución las operaciones correspondientes a cada instrucción son llevadas a cabo siempre y cuando no posean riesgo de tipo RAW. En caso de haberlo se retrasa la instrucción hasta eliminarlo.

3. La última etapa, la de escritura de registros, es la que sufre la mayor modificación, ya que la escritura es realizada en el CDB, el cual se encarga de llevar el resultado a los registros generales, estaciones de reserva o buffers según corresponda.

Es importante mencionar que si bien este algoritmo plantea ciertas mejoras tiene varias desventajas, siendo la principal que, como las instrucciones son emitidas de forma secuencial, las excepciones se disparan en el mismo orden en que ingresan al procesador, sin importar que estén siendo ejecutadas fuera de orden.

Este tipo de excepciones podrían ocurrir cuando no existe suficiente información del estado, los programas que experimentan excepciones precisas, en las que la instrucción que provocó la excepción puede ser determinada pueden evitar la detención. Sin embargo si se experimenta una excepción imprecisa el sistema no puede determinar la instrucción que provocó la excepción por lo que no se puede volver a ejecutar.

Al ocurrir una excepción precisa en el Pipeline se debe frenar el proceso y guardar el estado en el contador de programa, el cual es llamado PC por sus siglas en inglés, para que luego pueda ser restablecido de forma segura y correcta. Antes de restablecer una instrucción se deben tomar ciertas medidas para guardar el estado de forma adecuada, en primer lugar se debe insertar una instrucción trap[6], luego se deben eliminar las escrituras realizadas que causaron el fallo y también las hechas por las siguientes instrucciones, por último una vez terminada la eliminación se guarda el PC de la instrucción que ocasionó el problema. Si bien estos pasos se realizan para todas las instrucciones es importante remarcar que existen diferentes tipos de excepciones, como por ejemplo las sincrónicas las cuales ocurren cuando la localidad de datos y memoria es la misma. Este tipo de excepciones suelen encontrarse dentro de otras instrucciones, lo que genera que la instrucción deba ser parada y reiniciada. En este caso no

vemos la necesidad de recurrir a otro programa para guardar el estado de la ejecución del programa correctamente ya que es un proceso invisible al momento de ejecutar el programa, es decir, el pipeline provee la habilidad para que el procesador tome la excepción, guarde el estado y la reinicie sin afectar a la ejecución. Por otro lado están las excepciones asíncronas las cuales son causadas por el mal funcionamiento de dispositivos externos al CPU y memoria, esto genera que puedan ser manipuladas después de la finalización de la instrucción, si bien esto facilita su manejo si una excepción asíncrona ocurre dentro de una instrucción se debe finalizar el programa. [7]
Teniendo en cuenta todo lo descrito anteriormente podemos ver que entre el algoritmo del marcador y el algoritmo de Tomasulo hay varias diferencias, en primer lugar el método del marcador presenta cuatro etapas mientras que su sucesor tiene tres, una de las diferencias más evidentes es que el control del flujo en el scoreboarding es centralizado mientras que en el algoritmo de Tomasulo hace uso de las unidades funcionales y sus estaciones de reserva. Podemos hacer una distinción bastante notoria entre los algoritmos a la hora de manejar los riesgos de tipo WAR y WAW, donde uno utiliza el renombre de registros mientras que el otro hace uso de una memoria de registros para las instrucciones, esto en conjunto con la utilización del CDB para trasladar todos los resultados a las unidades funcionales son la principal diferencia entre ambos algoritmos. Si bien el algoritmo de Tomasulo es más eficiente a la hora de tratar riesgos ya que no frena la el flujo del programa en casos donde el algoritmo del marcador si, la complejidad del primero es mayor a la complejidad de este último.

5. TOMASULO EN PROCESADORES ACTUALES.

El algoritmo ideado por Robert Tomasulo es ampliamente utilizado, en esencia, en la actualidad. La implementación del algoritmo en las máquinas actuales se puede encontrar con distintas adaptaciones para mejorar su rendimiento y eficiencia, un ejemplo de esto es que antiguamente se aplicaba sólo para instrucciones largas mientras que actualmente las operaciones enteras también llevan planificación dinámica incluso veces las secciones FP e INT están separadas y otras en común. Las principales modificaciones que ha sufrido el algoritmo de Tomasulo son respecto a la predicción de dependencias y el uso de múltiples unidades funcionales. Esta última modificación permitió que se adopte en gran parte de los procesadores dado que los microprocesadores en la modernidad poseen varias unidades de las cuales cada una realiza diferentes tipos de operaciones, esto genera que se puedan ejecutar más instrucciones en simultáneo, lo que trae en consecuencia una mejora en el rendimiento por mayor probabilidad de presencia de ejecución fuera de orden.

Además de estas modificaciones generales, los microprocesadores modernos también suelen implementar modificaciones específicas para instrucciones o conjuntos de instrucciones particulares, pero en general las modificaciones al algoritmo de Tomasulo han permitido a los microprocesadores modernos alcanzar un mejor nivel de eficiencia y ejecutar conjuntos de instrucciones cada vez más complejos[5]. Este algoritmo se ha implementado en una amplia gama de microprocesadores actuales, desde procesadores de consumo hasta procesadores de alto rendimiento. Algunos ejemplos de gran importancia de implementaciones actuales incluyen el procesador Intel Core i9-12900K, que utiliza una variante del algoritmo de Tomasulo llamada TAGE (Thread Aware Global Execution), el procesador AMD Ryzen 9 7950X, que utiliza una variante del algoritmo de Tomasulo llamada RDNA3.

6. CONCLUSION.

Teniendo en cuenta todo lo detallado previamente, podemos afirmar que el informe revela que la ejecución fuera de orden, la planificación dinámica, el algoritmo del marcador y el algoritmo de Tomasulo son conceptos cruciales en la evolución de la eficiencia y el rendimiento de los procesadores. La ejecución fuera de orden, al permitir la ejecución simultánea de instrucciones que no posean una dependencia de datos con otras activas, genera una utilización más eficaz de los recursos disponibles. Por otro lado, la planificación dinámica sirve para adaptarse a las condiciones cambiantes presentes en la etapa de ejecución y lograr que la ejecución fuera de orden se haga sin comprometer la información. Es importante remarcar el algoritmo del marcador, el cual a pesar de haber sido desplazado por el algoritmo de Tomasulo, es un concepto fundamental al entender la evolución de la ejecución fuera de orden al pasar los años y como se ha logrado con cada modificación optimizar la ejecución de instrucciones mediante la gestión efectiva de los registros, contribuyendo así a la mejora del rendimiento general. Asimismo, el algoritmo de Tomasulo, caracterizado por sus mejoras en la predicción de dependencias y la implementación del bus común de datos ha desempeñado un rol trascendental en el incremento en el throughput y la ejecución eficiente de conjuntos de instrucciones en los microprocesadores contemporáneos.Estas innovaciones colectivas han logrado avances significativos, estableciendo nuevos estándares para la capacidad de procesamiento y la ejecución de operaciones en entornos computacionales modernos.

AGRADECIMIENTOS.

Ing. Marchi Edgardo, Ing Cervetto Marcos y Jáuregui Melina, docentes de cátedra Organización del Computador, Facultad de Ingeniería, Universidad de Buenos Aires.

REFERENCIAS.

[1] John L, Hennessy and David A. Patterson, “Pipelining: Basic and Intermediate Concepts” in Computer Architecture A Quantitative Approach, Morgan Kaufmann, 2019 [online].
Available:http://acs.pub.ro/~cpop/SMPA/Computer%20Architecture%20A%20Quantitative%20Approach%20(5th%20edition).pdf.
[2] A. S. Tanenbaum & H. Bos, "Hazards" in Modern Operating Systems, 4th ed, (2018). Pearson Education.
[3] A. Sebastian, ExtremeTech, (2012),
Available:https://www.extremetech.com/extreme/125271-the-history-of-supercomputers.
[4] IBM, “System/360 Model 91” IBM Archives, [online].
Available:https://www.ibm.com/ibm/history/exhibits/mainframe/mainframe_PP2091.html.
[5] Academia Lab. (2023). Set de instrucciones arquitectura. Enciclopedia. [online].
Available:.https://academia-lab.com/enciclopedia/set-de-instrucciones-arquitectura/
[6] Departamento de ciencias y computación, “Interrupciones”, Universidad

Nacional del Sur [online].
Available:https://cs.uns.edu.ar/~so/data/apuntes/SO-2020-mod%2002.pdf,
[7] Luis Fernando Ruiz juarez, “consideraciones de diseño e implementación del pipeline”, Universidad de San Carlos de Guatemala [online].
Available:http://biblioteca.usac.edu.gt/tesis/08/08_0117_EO.pdf.