

Reporte de Proyecto

Proyecto 1- Simulador del algoritmo de Tomasulo

David Felipe Duarte Sánchez, Wanderley Cortés Morales, Christian Arguedas Maceo

Resumen

Este reporte se presenta la implementación del algoritmo de Tomasulo para ejecutar ordenes fuera de orden ante un set de instrucciones con distintas dependencias de datos. Dicho algoritmo se representa por medio de un simulador diseñado en *Python* con el fin de mostrar su comportamiento ante distintas instrucciones y condiciones en el algoritmo ante distintas condiciones de entrada.

1. Diseño del simulador

El algoritmo de Tomasulo emplea el uso de un set de instrucciones entrantes al la estación de reservación, consecuente a esto, manda la instrucción necesaria para ser procesada por el ALU (operador en este caso) y posteriormente guarda el resultado una vez finalizado, caso de no ser posible espera que el registro del cual obtiene la información este disponible, de este modo evita los problemas de RAW(*read after write*), WAR(*write after read*) y WAW(*write after write*) permitiendo la ejecución fuera de orden sin inconvenientes empleando un bus de datos común[1][2].

Para el diseño del **simulador** del algoritmo de Tomasulo, el cual se puede observar su diagrama en la Fig. 1 se utilizo el lenguaje de programación Python. Las etapas en las cuales se dividió el programa se anotaron a continuación. En el subprograma llamado **Cola**, se gestiona la cola de instrucciones, además de esto se encarga de revisar si el destino de registro se encuentra disponible, de no ser así envía al reservation station la instrucción, por otro lado, de encontrarse disponible envía la instrucción para ser procesada. El subprograma **Reservation Station (RS)**, se encarga de administrar las operaciones que no se pueden realizar por que tienen algún valor de registro pendiente, el mismo se encarga de esperar los ciclos correspondientes (6 ciclos en caso de multiplicación o 4 ciclos en caso de suma o resta) y ejecutar la siguiente instrucción apenas se tenga el valor del registro requerido. Por otro lado en el subprograma **Common data bus**, se envían los resultados de las operaciones para que sean actualizados los valores de los registros y se cambie el estado del registro de ocupado a libre.

Por otro lado en el subprograma **Configuraciones**, es en donde se asigna el tamaño de las reservation stations tanto para las de **Suma y resta** como para la de **Multiplicación**, es de suma importancia hacer que las estaciones de reserva tengan valores lógicos racionales ya que un valor muy grande dará

resultados que no serán de mucha utilidad. Además de esto en la lista llamada **Ingresar nombre de la lista** se podrán colocar los valores iniciales de los registros. En los subprogramas llamados **Sumador** y **Multiplicador** es en los cuales se generan la mayor cantidad de operaciones ya que en ellos se generan las operaciones contenidas en las instrucciones. El subprograma de **Visualización**, es el encargado de mostrar los cambios que ocurren ciclo a ciclo en la consola del programa. Finalmente todos los subprogramas mencionados anteriormente son ejecutados en el programa llamado **Main**, el cual se ejecuta y comienza a generar los cálculos para cada ciclo de trabajo y detiene el programa cuando el mismo termina de ejecutar la ultima instrucción de la lista.

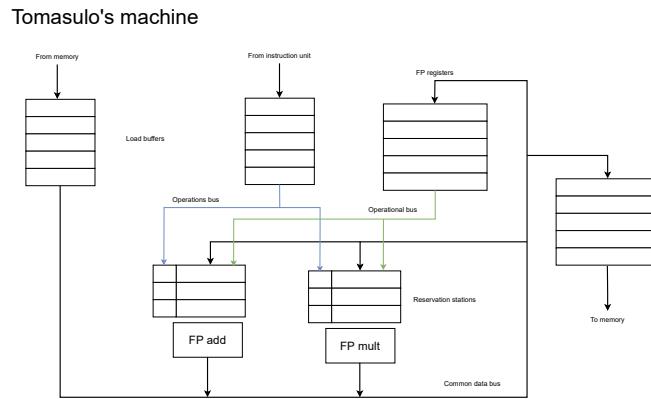


Figura 1: Máquina de Tomasulo IBM 360/91

2. Resultados

Los resultados obtenidos para este apartado se muestran a continuación, el código que se ejecuto para poner a prueba el simulador fue el siguiente conjunto de instrucciones. Llenando los registros inicialmente con el numero de registro +1, de forma en que el Registro **R0 = 1** hasta llegar al registro **R11 = 12**

- MUL R2, R0, R1
- ADD R4, R2, R3
- ADD R6, R1, R5
- ADD R9, R7, R8
- MUL R10, R6, R9
- ADD R4, R4, R10

Una vez introducido el conjunto de instrucciones en el archivo **Codigo_de_entrada.txt** se ejecuta el programa y se obtienen los ciclos de reloj mostrados en la Fig. 2.

3. Análisis de resultados

Como se puede observar en la Fig.2 se colocaron las imágenes correspondientes a cada ciclo de reloj que ejecuto el simulador para resolver la cola de instrucciones que se mostró anteriormente en este informe. Contrastando estos datos con los obtenidos en la presentación vista en clase se logra determinar que el simulador realizado cumple de forma satisfactoria la tarea. Cabe destacar que se realizaron varios experimentos, en el cual el primero de ellos. La **Primera prueba** se realizó con una reservation station para suma y resta de tamaño 2 y para multiplicación de tamaño 2 en el cual el programa tomaba 26 ciclos de reloj, la **segunda prueba** se realizó con una reservation station para suma y resta de tamaño 3 y para multiplicación de tamaño 2, analizando que la mayoría de instrucciones que se ejecutan son sumas, para esta prueba el programa tomó 21 ciclos de reloj. La **última prueba** se realizó con una reservation station para suma y resta de tamaño 2 y para multiplicación de tamaño 1 en el cual el programa no logró finalizar satisfactoriamente los ciclos de operación ya que ocupaba una reservation station de multiplicación para colocar la siguiente instrucción.

4. Conclusiones

Se determina que la configuración de las reservation stations mas favorable para el conjunto de instrucciones analizado es el de 2 para suma y resta y 2 para multiplicación. De esta forma se obtiene una buena relación entre la cantidad de ciclos de ejecución y la cantidad de RS que son necesarias. Por otro lado, se logra observar de una mejor manera como la ejecución fuera de orden de un código de entrada reduce la cantidad de ciclos de reloj que podría durar un programa ya que hace un uso dinámico de los mismos disminuyendo con esto la cantidad de ciclos que no serían aprovechados. Finalmente se determinó que un set de instrucciones puede ejecutarse de una forma mas efectiva haciendo uso de la mayor parte de los ciclos de reloj gracias a la ejecución dinámica que se logra gracias a las reservation stations y demás componentes necesarios para realizar este simulador.

Referencias

- [1] Hennessy, J.L., & Patterson, D.A. (2011). Computer architecture: A quantitative approach. Elsevier.
- [2] J.R "Tomasulo's Algorithm Take 3" YouTube, Oct. 12, 2018 [Video file]. Available: <https://www.youtube.com/watch?v=zS9ngvUQPNM&t=392s> [Accessed: Nov. 1, 2021].