

Laboratorio 2: Operaciones con números de punto flotante en Ensamblador MIPS

Reinaldo Pacheco Parra (14 horas)
 Departamento de Ingeniería Informática
 Universidad de Santiago de Chile, Santiago, Chile
 reinaldo.pacheco@usach.cl

Resumen—En el siguiente informe se documentará el procedimiento realizado para resolver una problemática mediante el uso de MARS Mips. Para ello, se debió realizar una investigación del funcionamiento de los números punto flotante con el fin de calcular una estimación del número π mediante un código aplicando el uso de subrutinas, funcionamiento de registros y saltos. Además se aplican temas tales como el uso de operaciones con números de punto flotante y repeticiones.

Palabras claves—MARS, MIPS, Saltos, Registros.

I. INTRODUCCIÓN

En el siguiente laboratorio se lleva a cabo la estimación del número π mediante un programa en lenguaje ensamblador construido en el simulador MARS (MIPS Assembler and Runtime Simulator). Para ello, se utilizarán operaciones de punto flotante y subrutinas. Este programa se construye con el propósito de abrir el portal generado por el científico Esteban Strange para traerlo de vuelta a su dimensión. Para lograrlo, es necesario tener conocimientos sobre algunos conceptos matemáticos como el cálculo de distancias y redondeo de números, así como habilidades en el uso de operaciones con punto flotante y generación de puntos aleatorios en MIPS. La estimación de π se mostrará a través de la consola para que pueda ser visualizada por el usuario y determinar si efectivamente, es correcta. Los principales objetivos de este laboratorio son: comprender y aplicar operaciones de punto flotante y subrutinas en MIPS, así como estimar el valor de π mediante la creación de un programa que simule el método de Monte Carlo en MIPS.

II. ANTECEDENTES

Mars MIPS es un entorno de desarrollo integrado (IDE) para el lenguaje de programación MIPS, que se utiliza en la arquitectura de procesadores MIPS. Este IDE permite escribir, depurar y ejecutar programas en un entorno gráfico de usuario, lo que lo hace útil tanto para estudiantes como para profesionales que trabajan con este simulador. [1]

El manejo de direcciones de memoria es fundamental en la programación en MIPS. Cada instrucción trabaja con una dirección de memoria que indica la ubicación específica de los datos que se deben procesar. Por lo tanto, es importante que el programador maneje adecuadamente e indique correctamente las direcciones de memoria a utilizar para evitar errores en el programa. [2]

Método de Monte Carlo: El método de Monte Carlo utiliza números aleatorios para estimar numéricamente el valor de π . Se generan puntos aleatorios en un cuadrado y se traza un círculo unitario dentro de él. Al contar los puntos dentro del círculo divididos en el total de puntos que se generaron, se puede obtener una aproximación de π multiplicando por 4 esa proporción. (ver Formula 1 en II-B.) [3]

Subrutina: Una subrutina es una sección de código que realiza una tarea específica y se puede llamar desde diferentes partes del programa. Las subrutinas ayudan a reducir la cantidad de código que se repite y a dividir el código en partes más pequeñas y manejables. [4]

Números Punto Flotante: En Mars MIPS, se refiere a la representación de números decimales. El simulador contiene instrucciones y registros para realizar operaciones aritméticas y lógicas con números de punto flotante. El uso de estos números es importante para realizar cálculos con mayor precisión. Algunas de las principales operaciones con punto flotante se muestran en la sección II-A. Tablas, en la Tabla I "Métodos y operaciones básicas con punto flotante del conjunto de instrucciones MIPS de Mars"

II-A. Tablas

Método/Operación	Función
f0 a f31	Son los espacios de almacenamiento en el procesador para punto flotante en MIPS
add.d	Suma de dos números de punto flotante
sub.d	Resta de dos números de punto flotante
c.eq.d	Compara si dos números de punto flotante son iguales
mul.d	Multiplicación de dos números de punto flotante y almacena el resultado en un registro punto flotante
sqr.d	Raíz cuadrada de un número de punto flotante y almacena el resultado en un registro punto flotante
ra	Es un registro que almacena la dirección de retorno de la llamada de una función.
sp	Puntero de pila, apunta al último elemento en la pila.
div.d	Divide dos números en punto flotante de 64 y guarda el resultado en punto flotante.
mtc1	Mueve un valor de un registro a un registro de punto flotante.

Tabla I: Métodos y operaciones básicas con punto flotante del conjunto de instrucciones MIPS de Mars [5]

II-B. Fórmulas

Se presenta la fórmula para calcular π mediante el método de Monte Carlo utilizada en este laboratorio y que fue creada con el editor

$$\pi = 4 \times \left(\frac{\text{puntos dentro del círculo}}{\text{puntos totales generados}} \right) \quad (1)$$

Donde la estimación de π corresponde a la multiplicación de 4 por la división entre la cantidad de puntos generados dentro del círculo dividido respecto a la cantidad total de puntos generados.

Otra de las formulas utilizadas durante este laboratorio es la de distancia o norma euclidiana, la cual está representada por:

$$\text{Distancia} = \sqrt{x^2 + y^2} \quad (2)$$

Donde en este caso, la componente x será el valor del primer número generado y la componente y será el valor del segundo número generado. El resultado de esta operación será un número punto flotante.

III. MATERIALES Y MÉTODOS

III-A. Materiales

Se debe descargar el simulador MARS Mips 4.5 de su página oficial para abrir los archivos con extensión .asm. Además, el dispositivo que se encarga de almacenar, procesar y ejecutar los códigos es un Notebook NITRO 5 con procesador Intel® Core™ i5-10300H CPU @ 2.50 GHz 16 GB de RAM (15,8GB utilizables), Sistema operativo de 64 bits, procesador x64 con una tarjeta gráfica NVIDIA® GeForce® GTX 1650 y la versión 22H2 de Windows 11.

III-B. Métodos

El método para la construcción del programa es el siguiente:

1. **Generación de punto:** Se genera un punto aleatorio (x, y) dentro de un cuadrado delimitado por los valores de 0 a 100 en ambas coordenadas.
2. **Construcción de la distancia del punto:** Se calcula la distancia del punto generado en el paso anterior al origen $(0, 0)$ utilizando la fórmula del módulo o norma Euclidiana (ver formula 2 en II.B)
3. **Determinación del punto:** Se verifica si la distancia calculada en el paso anterior es menor a 100. Si la distancia es menor a 100, significa que el punto se encuentra dentro del círculo de radio 100; de lo contrario, el punto está fuera del círculo.
4. **Cantidad de puntos dentro:** Se mantiene un contador para registrar la cantidad de puntos generados que se encuentran dentro del círculo. Cada vez que un punto cumple la condición de estar dentro del círculo, se incrementa el contador.
5. **Estimación de pi:** Utilizando la relación entre el área del círculo y el área del cuadrado, se puede obtener una estimación de pi. (ver fórmula 1 en II.B)
6. **Mostrar el resultado:** Finalmente, se muestra por consola la aproximación de π obtenida en el paso anterior.

IV. RESULTADOS

Los resultados de la experiencia consisten en la estimación del número π mediante la generación de puntos. El programa fue ejecutado con diferentes cantidades de puntos para realizar dicha estimación con el fin de analizar el comportamiento del resultado.

En primer lugar, se utilizó el programa con 100 puntos generados por MIPS (ver Figura 1). En esta configuración, se observa que la estimación no es muy cercana al valor original, presentando un margen de error de hasta 0.3 décimas.

Posteriormente, se modificó el programa para ejecutarlo con 1000 puntos generados por MIPS, lo cual resultó en una aproximación mucho más precisa al valor original (ver Figura 2).

Finalmente, se estimó el valor de π utilizando 10.000 puntos (ver Figura 3). En cada iteración, los resultados fueron aún más cercanos a los anteriores, presentando un margen de error menor.

En cuanto al análisis, se puede concluir que la estimación del número π es aceptable con 100 puntos, pero a medida que se incrementa la cantidad de puntos generados, el margen de error se reduce, obteniendo una aproximación más precisa. Es importante destacar que a medida que se aumenta la cantidad de puntos, el tiempo de respuesta del programa también aumenta debido a la mayor cantidad de iteraciones necesarias. Por lo que, si se modifica excesivamente el número de puntos, se puede obtener como resultado una respuesta lenta o un mal funcionamiento del programa.

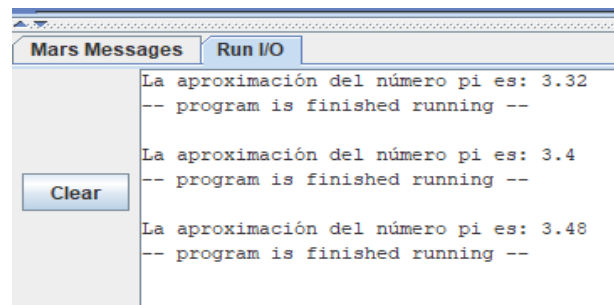


Figura 1: Estimación de π con 100 puntos

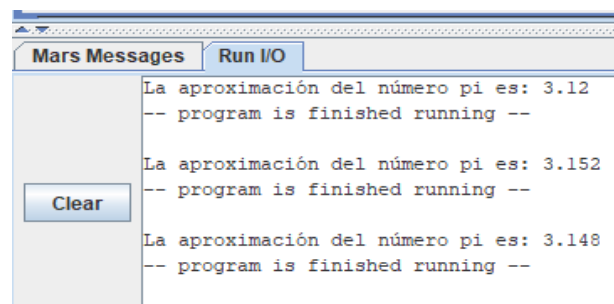


Figura 2: Estimación de π con 1000 puntos

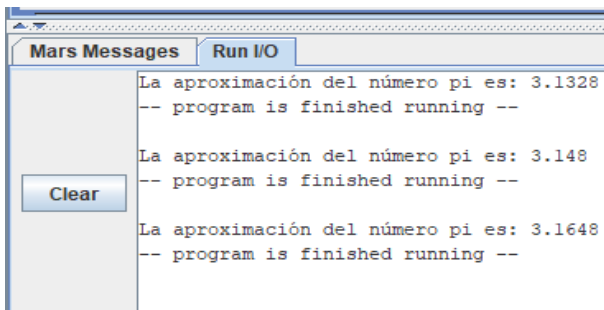


Figura 3: Estimación de π con 10.000 puntos

V. CONCLUSIONES

En conclusión, fue necesario para explorar y comprender los conceptos de números de punto flotante y el uso de subrutinas. Esto permitió desarrollar la comprensión de estos conceptos para realizar operaciones en el programa.

En esta ocasión, el laboratorio presentó una mayor variedad de operaciones disponibles, lo que facilitó la construcción de cálculos que hubieran sido complicados de realizar en el entorno de MIPS. Algunos ejemplos de estas operaciones fueron el redondeo de números y el cálculo de la raíz cuadrada utilizando el Método de Newton-Raphson.

Además, a partir del análisis, se pudo apreciar que a medida que el programa realiza más repeticiones, el tiempo de espera para obtener el resultado puede aumentar significativamente, aunque con el efecto de obtener un menor margen de error.

Finalmente, se lograron los objetivos establecidos inicialmente, que eran comprender y aplicar las operaciones de punto flotante. Estas operaciones fueron empleadas con éxito en la elaboración del programa, lo que contribuyó a alcanzar otro objetivo planteado, que era obtener una aproximación cercana al valor original de π .

REFERENCIAS

- [1] C. W. Kann, "Introduction to mips assembly language programming," [Online] <https://cupola.gettysburg.edu/cgi/viewcontent.cgi?article=1001&context=oer>, 2003, visitada el 26 de Abril del 2023.
- [2] A. H. Cerezo, "Arquitectura mips trabajo 2," [Online] https://www.infor.uva.es/~bastida/OC/TRABAJO1_MIPS.pdf, 2018, visitada el 22 de Mayo del 2023.
- [3] C. P. R. y George Casella, "Monte carlo statistical methods," [Online] https://mcube.lab.nycu.edu.tw/~cfung/docs/books/robert2004monte_carlo_statistical_methods.pdf, 2004, visitada el 14 de Junio del 2023.
- [4] D. Sweetman, "See mips run, segunda edición," [Online] <https://doc.lagout.org/electronics/doc/mips/See%20mips%20run.pdf>, 2006, visitada el 14 de Junio del 2023.
- [5] K. L. Algara, "Mars mips assembler and runtime simulator," [Online] <https://usermanual.wiki/Document/MarsManualUsuario.686917485/view>, 2012, visitada el 22 de Mayo del 2023.