UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Laboratorio 2. Acercándose al Hardware: Programación en Lenguaje Ensamblador

Integrantes:	Florencia Corvalán Lillo
Curso:	Laboratorio Organización de computadores
Sección	L-2
Profesor(a):	Leonel Medina
Ayudante:	Alexander Palma

20 de Junio de 2020

Tabla de contenidos

1. Introducción

En este informe se presenta el desarrollo del segundo proyecto de laboratorio de la asignatura Organización de computadores, el que consiste en la implementación de programas en lenguaje ensamblador MIPS mediante el IDE MARS.

1.1. Objetivos

Los objetivos del presente proyecto se asocian al uso de MARS para implementar, ensamblar y depurar programas en lenguaje MIPS; a la utilización de instrucciones aritméticas, de salto y memoria; a entender el uso de subrutinas en este lenguaje, incluyendo el manejo del stack; a la realización de llamadas de sistema mediante "syscall"; y a la implementación de algoritmos para la resolución de problemas matemáticos.

1.2. Exigencias del proyecto

La primera parte del proyecto exige la implementación de un programa que determine el máximo entre dos números enteros ingresados por consola, y entrege el resultado por consola.

La segunda parte consiste en la implementación de programas que realicen cálculos de operaciones matemáticas para números enteros: multiplicación, factorial y división; esto se debe hacer sin utilizar las instrucciones propias de MIPS de multiplicación, división, desplazamiento, etc. Los operandos deben estar determinados dentro de la implementación y el resultado se debe entregar por consola. La división debe entregar el resultado con dos decimales.

Y la tercera parte consiste en la implementación de programas que realicen aproximaciones de funciones matemáticas: función seno, función seno hiperbólico y función logaritmo natural; mediante series de Taylor en torno a cero y de orden 7 o superior. Estas aproximaciones se deben implementar para números enteros no negativos y utilizando las operaciones matemáticas implementadas en la segunda parte. Los programas deben recibir por consola el número para el que se desea aproximar la función y entregar el resultado por consola con dos decimales.

1.3. Organización del documento

Primero se presentará un marco teórico en el que se definirán los conceptos y consideraciones sobre el proyecto. Luego se explicarán brevemente las soluciones implementadas y se describirá como es que se llegó a estas. Después se analizarán los resultados de las implementaciones, relacionando entradas con la cantidad de instrucciones ejecutadas. Por último, se presentaran conclusiones sobre el proyecto.

2. Marco teórico

A continuación se definen los conceptos y consideraciones que se utilizaron para un mejor entedimiento de lo expuesto en este informe.

2.1. Series de Taylor

Las series de Taylor se utilizaran en torno a cero y del orden tal que sea posible el calculo del factorial.

Serie de Taylor de la función seno (UCTemuco, 2015):

$$f(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2 \cdot n + 1)!} \cdot X^{2 \cdot n + 1} \tag{1}$$

Serie de Taylor de la función seno hiperbólico (Unknow, sf):

$$f(x) = \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1)!} \cdot X^{2 \cdot n + 1} \tag{2}$$

Serie de Taylor de la función logaritmo natural (MateFacil, 2016):

$$f(x + 1) = \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{n} \cdot X^{n} \tag{3}$$

2.2. Consideraciones

Se debe tener en consideración que el número máximo representable sin signo en MIPS, es decir, en 32 bits, es 4294967296. Es por eso que se determina el orden de cada serie de Taylor, siendo este el máximo posible, según el factorial máximo que se puede calcular, que es 12. Por lo tanto, para la serie de Taylor del seno y del seno hiperbólico se considera el orden de 5 como máximo para que el factorial esté dentro de los valores posibles. Para la serie de Taylor del logaritmo natural, el orden puede ser un número mayor a 5 porque no incluye un factorial, determinandose un orden de 7 dadas las exigencias del proyecto. También se debe considerar que para algunas instrucciones de MIPS no se produce overflow pero si se producen resultados erróneos, es por esto que para algunos casos, aunque los cálculos estén bien hechos y se ejecuten correctamente, el resultado puede ser incorrecto.

3. Desarrollo

En esta sección se explicarán brevemente las soluciones implementadas para los problemas descritos anteriormente. Y se describirán y analizarán los resultados obtenidos.

3.1. Soluciones

Debido a que, en todas las partes del proyecto, los argumentos de las subrutinas se deben entregar en determinados registros y que al hacer los saltos a las subrutinas se modifica la dirección de retorno es necesario ir guardando en el stack el valor de la dirección de retorno y los valores que se seguiran utilizando en cada llamado a una sub rutina y luego restaurar sus valores.

3.1.1. Parte 1

Primero se revisa si los numeros ingresados son iguales, si lo son se informa esto. Si no son iguales se llama a la subrutina calculoMaximo que determina cual de los numeros es el maximo, comparandolos y seleccionando el mayor, para luego imprimir el resultado.

3.1.2. Parte 2

La multiplicación (parte 2.a) se realiza mediante sumas repetitivas y un acumulador. El procedimiento ocurre en una subrutina llamada procedimientoMult en la que se llama a otras subrutinas para realizar el cálculo. Primero, mediante una subrutina se determina el signo del resultado de la multiplicación. Otra subrutina posiciona los operandos de manera que las sumas repetitivas se hagan en la menor cantidad de instrucciones. Después otra subrutina genera un ciclo que realiza las sumas y las acumula. Luego se llama a una subrutina que le aplica el signo al resultado. Finalmente se retorna a donde se llamo a procedimientoMult y se imprime el resultado.

El factorial (parte 2.b) se calcula por medio de multiplicaciones repetitivas, un acumulador y un contador, utilizando la operación de multiplicación de la parte 2.a. Mediante una subrutina llamada procedimientoFact se realizan los distintos procedimientos a través de otras subrutinas. Mediante una subrutina se efetúa un ciclo realizando las multiplicaciones

correspondientes y acumulando el resultado en un registro. Cuando se tiene el resultado se retorna a donde se llamo a procedimientoFact y se imprime.

La división (parte 2.c) se realiza por medio de restas repetitivas y un contador. La subrutina procedimientoDiv es la que lleva a cabo el procedimiento mediante otras subrutinas. La parte decimal se calcula con el resto de la división entera amplificandolo por 100, utilizando la multiplicación de la parte 2.a, y dividiendolo nuevamente, pero esta vez con registros de tipo double, sumando 0.01 a un acumulador cada vez que se hace una resta. Luego se castea el resultado de la parte entera a double y se suman a la parte decimal calculada. Cuando se tiene el resultado completo se retorna a donde se llamo a procedimientoDiv y se imprime el resultado. Esta operación sólo sirve para números enteros no negativos. Cabe mencionar que cuando la diferencia del resto de la división amplificado por 100 y el denominador es menor a cero, significa que los dos primeros decimales son cero, por lo tanto se determina esto sin hacer otro cálculo.

3.1.3. Parte 3

Esta parte corresponde a las implementaciones de las aproximaciones de las funciones Seno (parte 3.1), Seno hiperbólico (parte 3.2) y logaritmo natural (parte 3.3). Estas implementaciones siguen la misma lógica, lo que varía son los cálculos a realizar. En primer lugar se pide por consola el número del cual se quiere aproximar la función. Luego a través de una subrutina llamada, dependiendo de la función, calcularSeno, calcularSenoHip o calcularLogaritmoNatural, se ejecuta la aproximación por medio de otras subrutinas, realizando cada una un cálculo de una de las partes de la fórmula correspondiente. Se implementó una subrutina para realizar la potencia de un número, que utiliza la instrucción mul de MIPS, pero para el resto de operaciones de multiplicación y división se utilizan solo las implementadas en la parte 2 del proyecto. Como la división solo soporta números enteros no negativos como entradas, para determinar el signo del resultado de cada iteración, en el caso de la función seno y logartimo natural, se hace un cambio de signo a través de una resta de cero menos el resultado de la iteración dependiendo de cual sea el resultado de la potencia del -1.

3.2. Resultados

3.2.1. Parte 1

Las pruebas efectuadas se realizaron con distintas entradas: ambos números negativos, ambos números positivos, el primero positivo y el segundo negativo, y viceversa; con números entre -100000 y 1000000. Siendo los resultados bastante satisfactorios, ya que para todas las entradas probadas funcionó correctamente y entregó el resultado correcto. Se obtuvo un promedio de 32 instrucciones en total ejecutadas y los promedios de porcentajes de instrucciones por tipo de operanción presentados en el cuadro 1.

ALU	Jump	Branch	Memory	Other
44	9	6	0	41

Cuadro 1: Promedio de porcentajes Parte 1

3.2.2. Parte 2

Los resultados obtenidos en la parte 2.a, la implementación de la operanción multiplicación, se encuentran dentro de lo esperado. Esta es capaz de operar números enteros positivos, negativos y el cero; realiza los cálculos correctamente, entregando los resultados correctos. Se efectuaron 10 pruebas, con distintos números enteros negativos y positivos entre 1000000 y -1000, obteniéndose un promedio de 440.8 instrucciones totales ejecutadas por multiplicación, con la menor cantidad de instrucciones de 62 para -38x5 y la mayor de 1445 para 350x-1000; y los promedios de porcentajes de instrucciones por tipo de operanción que se muestran en el cuadro 2.

ALU	Jump	Branch	Memory	Other
61.6	7.3	21.3	3.1	6.7

Cuadro 2: Promedio de porcentajes Parte 2.a

La implementación de la operación factorial puede operar todos los números enteros entre 0 y 12, incluidos, obteniéndose resultados correctos para todos estos casos. El factorial de 13 excede al número máximo sin signo representable en 32 bits, es por esto que

desde el 13 en adelante no se puede realizar el cálculo del factorial con esta implementación. Se efectuaron 13 pruebas, con los números del 0 al 12, y se obtuvo un promedio de 394 instrucciones totales ejecutadas por factorial, con la menor cantidad de instrucciones de 22 para el factorial de 0 y la mayor de 856 para el factorial de 12; y los promedios de porcentajes de instrucciones por tipo de operanción que se muestran en el cuadro 3.

ALU	Jump	Branch	Memory	Other
38	19.5	15.8	15.7	11

Cuadro 3: Promedio de porcentajes Parte 2.b

Para la implementación de la operación división se obtuvieron resultados regulares. En primer lugar, está implementada para operar sólo números enteros no negativos, lo que limita bastante su uso. Por otra parte, cuando la diferencia entre el numerador y el denominador es muy grande, el tiempo de ejecución y el número de instrucciones ejecutadas incrementa considerablemente, debido a que aumentan las restas a realizar. Por último, el hecho de tener que multiplicar por 100 el resto de la división para obtener el resultado de los decimales hace que aumente la cantidad de instrucciones ejecutadas y puede limitar su uso al tener que representar el resto como un número más grande. Sin embargo, todos los resultados de las pruebas realizadas fueron correctos. Se efectuaron 10 pruebas, con numeradores entre 1 y 10000 y denominador 7. Se obtuvo un promedio de 1368 instrucciones totales ejecutadas por división, con la menor cantidad de instrucciones de 152 para 1/7 y la mayor de 6050 para 10000/7; y los promedios de porcentajes de instrucciones por tipo de operanción presentados en el cuadro 4.

ALU	Jump	Branch	Memory	Other
38.9	5.1	22	3.5	30.5

Cuadro 4: Promedio de porcentajes Parte 2.c

3.2.3. Parte 3

Los resultados de esta parte son regulares. Se tiene que para la aproximación del seno y del seno hiperbólico, solo se pueden aproximar correctamente números del 0 al 7, lo que

es una limitación bastante grande. Al tener un orden 5 para la series de Taylor, la potencia máxima a calcular es el número ingresado elevado a 11, y 8 elevado a 11 es 8589934592 que es mayor a 4294967296, que es el máximo número sin signo representable en 32 bits. Es por esto que al intentar aproximar el seno o seno hiperbólico de 8, las implementaciones entregan resultados erróneos. Por último, como la división solo entrega dos decimales se pueden producir errores de precisión.

Para la implementación de la aproximación de la función seno se realizaron 8 pruebas, aproximando desde el número 0 hasta el 7, los resultados fueron correctos para los números del 0 al 5, en la aproximación del 6 y el 7 el segundo decimal no es el correcto. Para las pruebas realizadas se obtuvo un promedio de 5474.8 instrucciones totales ejecutadas por aproximación, siendo la menor cantidad de instrucciones de 3306 para 0 y la mayor de 7480 para 7; y los promedios de instrucciones por tipo de operación son los del cuadro 5.

ALU	Jump	Branch	Memory	Other
43	14.1	18.3	12.1	12.5

Cuadro 5: Promedio de porcentajes Seno

Para la implementación de la aproximación de la función seno hiperbólico se realizaron las mismas pruebas que para la de la función seno. Los resultados de las pruebas realizadas no son totalmente correctos ya que el segundo decimal es una unidad menor de lo que debería ser, excepto para la aproximación de cero. Se obtuvo un promedio de 5223.7 de instrucciones totales ejecutadas por aproximación, con la menor cantidad de instrucciones de 3055 para 0 y la mayor de 7229 para 7; y los promedios de instrucciones por tipo de operación se encuentran en el cuadro 6.

ALU	Jump	Branch	Memory	Other
43.3	14	18.8	11.3	12.5

Cuadro 6: Promedio de porcentajes Seno hiperbólico

Para la aproximación del logartimo natural se tiene que al ser implementado con un orden de 7, debería funcionar hasta el número 23, pero falla en una multiplicación de la subrutina que realiza la potencia, que está implementada con la instrucción mul de MIPS.

Se realizaron pruebas para los enteros del 0 al 7, incluidos, y se obtuvo resultados correctos, excepto por la aproximación de 4 y de 6, en las que el segundo decimal no es el correcto. Se obtuvo un un promedio de 106607.1 instrucciones totales ejecutadas por aproximación, con la menor cantidad de instrucciones de 854 para 0 y la mayor de 567309 para 7; y los promedios de instrucciones por tipo de operación resumidos en el cuadro 7. El gran incremento en la cantidad de instrucciones ejecutadas a medida que incrementa el número a aproximar se debe a que la diferencia entre el numerador y el denominador en cada iteración es mucho mayor a medida que el número a aproximar es mayor, haciéndose mucho más larga la operación de división.

ALU	Jump	Branch	Memory	Other
41.5	5.7	20.6	7.1	25.2

Cuadro 7: Promedio de porcentajes Logaritmo natural

La comprobación de los resultados de las aproximaciones de las tres funciones se realizaron comparando el resultado entregado por la implementación con el resultado correspondiente a la sumatoria con el orden determinado. Es por esto que cuando se dice que los resultados son correctos, lo son para esta forma de implementación y no necesariamente corresponden a los valores reales de la función, ya que las series de Taylor realizan aproximaciones y su nivel de precisión depende de su orden, mientras mayor el orden mayor precisión, y por las explicaciones ya mencionadas no es posible tener un orden muy grande.

Para obtener el número de instrucciones totales y los porcentajes de instrucciones por tipo de operación se utilizó la herramienta Instruction Statistics de MIPS.

4. Conclusiones

Se puede observar que las implementaciones realizadas están limitadas principalmente por la capacidad que tiene MIPS de 32 bits por palabra. Esto provoca que la implementación del factorial soporte una cantidad pequeña de entradas, y que la aproximación de la función seno y seno hiperbólico estén limitadas al cálculo correcto de muy pocas entradas. Una de las cosas que se podría mejorar es la implementación de la división, en la que incrementa considerablemente la cantidad de instrucciones ejecutadas y el tiempo de ejecución a medida que aumenta la diferencia entre el numerador y el denominador, haciendo que retrase y que aumente la cantidad de instrucciones ejecutadas en las implementaciones donde se utiliza; y se podría modificar para que sea capaz de soportar entradas negativas. Por otra parte, se encuentran las imprecisiones en los decimales, que se debe a que la implementación realizada sólo considera los dos primeros decimales para todos los procedimientos. Cabe destacar que a pesar de lo mencionado anteriormente, se lograron implementaciones funcionales que entregan resultados correctos para ciertas entradas.

Para concluir, se puede decir que se cumplieron los objetivos del proyecto, ya que se logró la implementación funcional de todas sus partes, mediante el IDE MARS utilizando instrucciones aritméticas, de salto y memoria, y a través de subrutinas y el manejo correcto del stack.

5. Anexos

5.1. Gráficos parte 3

A continuación se muestran los gráficos asociados a la parte 3 del proyecto, donde se muestran las relaciones entre entrada y número de instrucciones totales ejecutadas para cada una de las aproximaciones de funciones. En la figura 1 se muestra el de la aproximación del seno, en la figura 2 del seno hiperbólico y en la figura 3 del logaritmo natural.

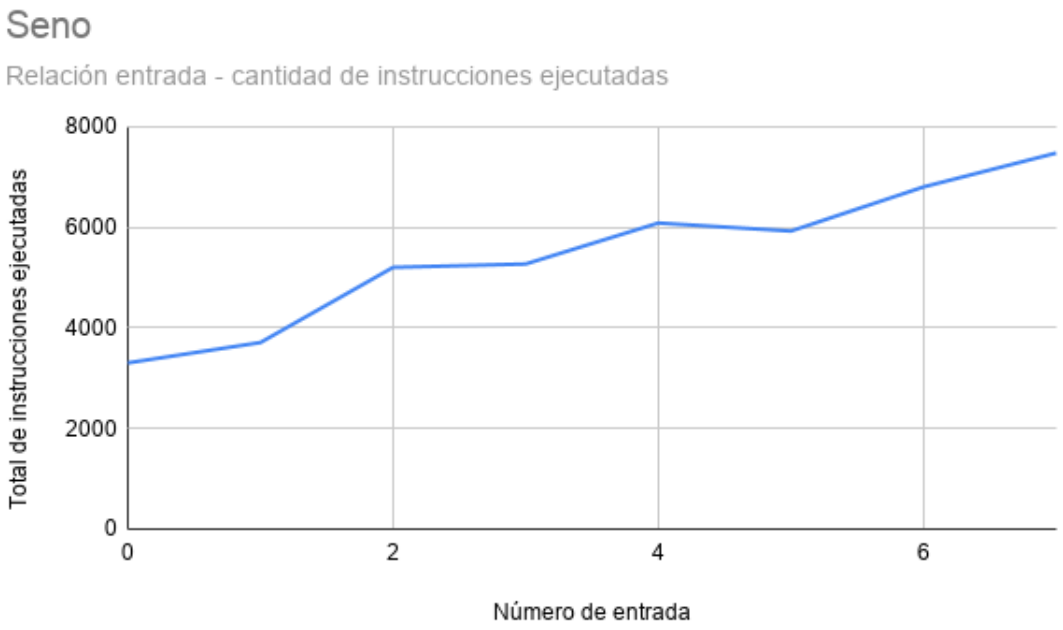

Figura 1: Gráfico aproximación función seno

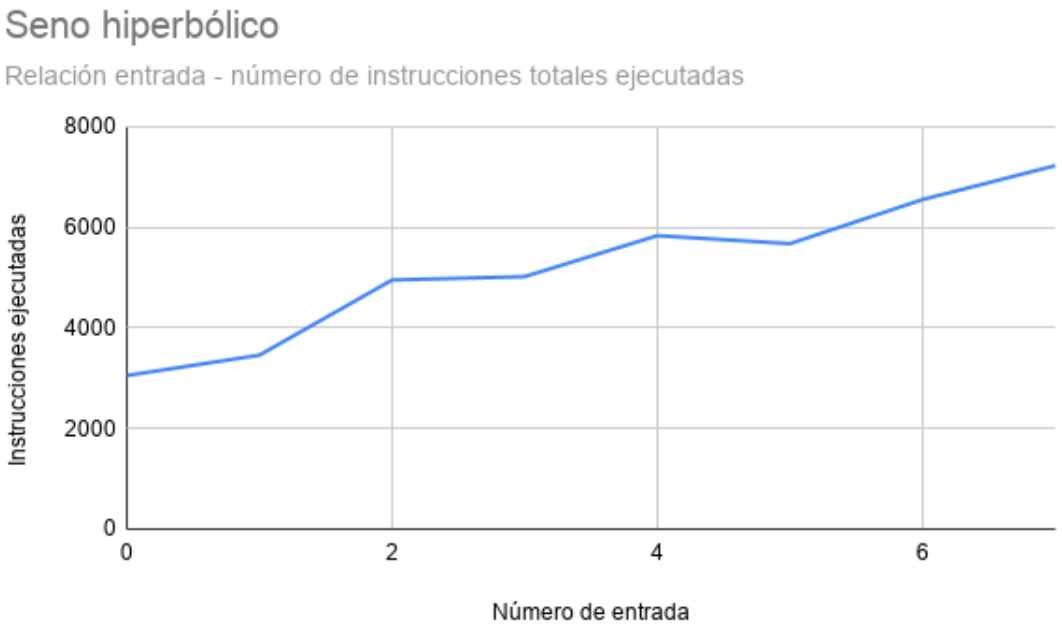

Figura 2: Gráfico aproximación función seno hiperbólico

Figura 3: Gráfico aproximación función logaritmo natural

Bibliografía

MateFacil (2016). Serie de taylor (maclaurin) del logaritmo natural. [Online] https://www.youtube.com/watch?v=_NIVO-TmO2U.

UCTemuco (2015). Seno y coseno en series de tayor aplicados en lenguaje python. [Online] http://repositoriosmichelmunoz.blogspot.com/2015/10/seno-y-coseno-en-series-de-taylor-para.html.

Unknow (s.f.). Cálculo del seno hiperbólico mediante la serie de taylor. [Online] https://programacion-ansi-c.blogspot.com/2012/03/calculo-del-seno-hiperbolico-mediante.html.