Practica 1 - Seno y Coseno

Henry José, Chacón Barillas, 202002535

Escuela de Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos de Guatemala

La práctica consiste en realizar un programa el cual pueda calcular la serie seno y coseno de un número.

I. OBJETIVOS

- Que el estudiante sea capaz de realizar distintas operaciones matemáticas mediante un programa utilizando lenguaje ensamblador.
- Que el estudiante domine los principales Mnemónicos para utilizarlos correctamente en las siguientes prácticas.

II. MARCO TEÓRICO

A. Series de Taylor

Las series de Taylor se utilizan para aproximar funciones que pueden ser complejas o no tienen una representación analítica simple. Se usan ampliamente en cálculo, física, ingeniería y muchas otras áreas para:

- Aproximaciones: Se pueden usar los primeros términos de la serie para obtener una aproximación de la función.
- Solución de ecuaciones diferenciales: A menudo se utilizan series de Taylor para resolver ecuaciones diferenciales de forma aproximada.
- Análisis de errores: Las series de Taylor permiten estimar cuán precisa es la aproximación de una función en términos de los primeros términos de la serie.

Las series de Taylor para las funciones de seno y coseno son ejemplos clásicos que muestran cómo se puede aproximar una función mediante una serie infinita. Estas series se suelen desarrollar en torno al punto a=0, lo que las convierte en series de Maclaurin.

$$\operatorname{sen} x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}\, x^{2n+1} \quad \textbf{para toda } x$$
$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!}\, x^{2n} \quad \textbf{para toda } x$$

Figura 1: Forma general de la serie respectiva de Taylor para el seno y coseno.

B. Keil μVision

Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) ampliamente utilizado para el desarrollo de software embebido, especialmente en microcontroladores basados en la arquitectura ARM. Fue desarrollado por Keil, una empresa que ahora es parte de ARM Holdings, y es uno de los entornos de desarrollo más populares para programar y depurar microcontroladores.

Caracteristicas:

- Soporte de Microcontroladores: Keil uVision es compatible con una amplia gama de microcontroladores basados en ARM, como los de la serie Cortex-M, y también soporta otros microcontroladores como los basados en 8051.
- Compilador ARM (ARMCC): Incluye el compilador de ARM, que optimiza el código para la arquitectura ARM, permitiendo un rendimiento eficiente y un uso reducido de memoria.
- Depuración y Simulación: Ofrece herramientas avanzadas de depuración y simulación que permiten al usuario probar y depurar el código en tiempo real o a través de simulaciones de hardware.
- Editor de Código: Tiene un editor de código integrado con soporte para sintaxis destacada, autocompletado, y herramientas de búsqueda que facilitan la escritura y edición de código.
- Gestión de Proyectos: Permite la gestión de proyectos con múltiples archivos, automatización de tareas de compilación y enlace, y configuración de opciones de compilación.
- Configuración de Periféricos: Para microcontroladores compatibles, uVision incluye herramientas gráficas que permiten configurar fácilmente los periféricos, pines, y otros recursos del microcontrolador.
- Soporte para Lenguaje Ensamblador y C/C++: uVision permite escribir código en ensamblador, C y C++, lo que brinda flexibilidad en la programación y optimización del código para aplicaciones embebidas.

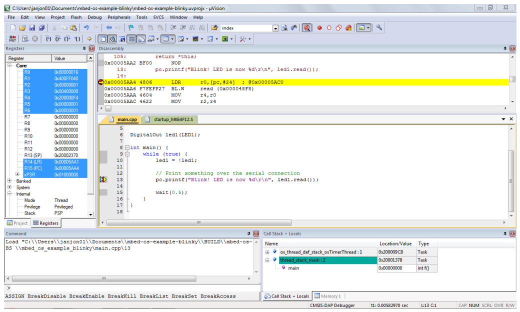

Figura 2: Entorno de Keil μVision

III. DIAGRAMA DE FLUJO

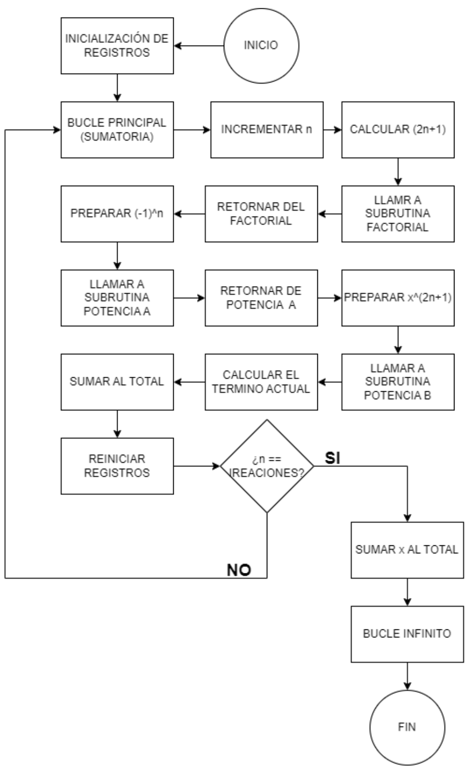

Figura 3: Algoritmo del programa realizado

IV. DISEÑO EXPERIMENTAL

El objetivo del código es calcular la suma de los primeros n términos de una serie de Taylor que aproxima una función trigonométrica. El código realiza este cálculo utilizando registros de coma flotante en un procesador ARM.

A. Materiales

- Laptop
- Keil μVision

B. Procedimiento

1. *Inicialización de Registros*

Al comienzo del código, se cargan valores iniciales en varios registros:

- Ángulo en radianes (S0): Se carga el valor de x (0.3926990817 radianes), que corresponde a 22.5 grados, el cual se va a usar para evaluar la serie.
- Número de iteraciones (S6): Se establece en 50, indicando que se calcularán 50 términos de la serie.
- Otros registros (S2, S3, S4, S5, S7, S9, S10, S11, S13, S16): Se inicializan a valores como 1.0 o 2.0, que se utilizarán a lo largo de los cálculos.

2. *Bucle Principal (Sumatoria)*

El bucle principal se encarga de calcular cada término de la serie y sumarlo al total acumulado.

- Incrementar el Contador de Iteraciones (S4): En cada iteración, el contador n se incrementa (VADD.F32 S4, S7).
- Calcular el Denominador (2n + 1): se multiplica n por 2 (VMUL.F32 S8, S4, S3) y se le suma 1 (VADD.F32 S8, S7). Este valor se almacena en el registro S15.
- Llamada a Subrutina Factorial: Se llama a una subrutina que calcula el factorial de 2n + 1. El resultado se almacena en el registro S2.
- Calcular la Potencia $(-1)^n$: Se prepara el valor para calcular $(-1)^n$ (VADD.F32 S12, S4, S13), y se llama a la subrutina potenciaa, que realiza la multiplicación repetida hasta que S12 llega a 0. El resultado se almacena en S11.
- Calcular la Potencia $x^{(2n+1)}$: Se prepara el valor S15 y se llama a la subrutina potenciab, que calcula $x^{(2n+1)}$ mediante multiplicación repetida. El resultado se almacena en S16.
- Calcular el Término Actual de la Serie: El término actual de la serie se calcula como:

$$terminoactual = \frac{(-1)^n * x^{2n+1}}{(2n+1)!} \tag{1}$$

Se divide $(-1)^n$ entre $(2n+1)!$ (VDIV.F32 S18, S14, S2) y luego se multiplica por $x^{(2n+1)}$ (VMUL.F32 S18, S17).

- Acumular el Término al Total: El valor calculado se suma al acumulado (VADD.F32 S9, S18).
- Reiniciar Registros: Los registros S2, S16, y S11 se reinician a 1.0 para estar listos para la siguiente iteración.
- Verificar si se Han Completado Todas las Iteraciones: Se compara n con el número total de iteraciones (S6). Si no se han completado, se regresa al inicio del bucle para calcular el siguiente término. Si se han completado todas, se continúa al siguiente paso.

3. Finalización de la Sumatoria

Después de completar las iteraciones:

- Se suma el valor inicial del ángulo (x) al total acumulado (VADD.F32 S9, S0). Este paso asegura que la suma final incluye todos los términos hasta n y añade el ángulo base.

4. Bucle Infinito

Para mantener el programa en ejecución después de completar el cálculo:

- El código entra en un bucle infinito (Loop: B Loop). Este bucle asegura que el programa no termine y permite mantener el resultado en los registros.

V. CÓDIGOS

A. Código para el calculo del seno

```

;El presente codigo calcula la funcion seno de cualquier angulo medido en radianes por medio de la serie de Taylor

  AREA codigo, CODE, READONLY, ALIGN=2
  THUMB
  EXPORT Start

Start
  VLDR.F32 S0, = 0.3926990817 ;Angulo del que se desea determinar el seno
  VLDR.F32 S6, = 50 ;Cantidad de iteraciones a realizar
  VLDR.F32 S2, = 1
  VLDR.F32 S3, = 2  ;Constante
  VLDR.F32 S4, = 0  ;Numero de iteracion actual (n)
  VLDR.F32 S5, = -1 ;Constante
  VLDR.F32 S7, = 1  ;Constante
  VLDR.F32 S9, = 0  ;Almacena la sumatoria, es decir el total
  VLDR.F32 S10, = 3
  VLDR.F32 S11, = 1
  VLDR.F32 S13, = 0 ;Constante
  VLDR.F32 S16, = 1

sumatoria
  VADD.F32 S4, S7   ;Aumentamos el numero de la iteracion
  VMUL.F32 S8, S4, S3 ;Obtenemos el valor de 2n para el denominador y lo almacenamos en S8
  VADD.F32 S8, S7   ;Complementamos la expresion del denominador (2n+1)    VADD.F32 S8, S7 ---------->>>>>>>>>PARA COSENO SUSTITUIR POR "VADD.F32 S8, S13"
  ;VADD.F32 S8, S13 ;<<<-----COSENO
  VADD.F32 S15, S8, S13 ;Se almacena nuevamente el valor del denominador (2n+1) ahora en S15
  BL factorial    ;Salta a la funcion de factorial
  VADD.F32 S12, S4, S13 ;Se asigna el valor de n a S12 para la funcion potenciaa
  BL potenciaa
  VADD.F32 S14, S11, S13  ;Se asigna a S14 la potencia del numerador
  BL potenciab
  VADD.F32 S17, S16, S13  ;Se asigna a S17 la segunda potencia del numerador
  VDIV.F32 S18, S14, S2 ;Se divide el numerador (-1)^n de S14 entre el factorial (2n+1)! en S2
  VMUL.F32 S18, S17 ;Se multiplica la division actual por la segunda potencia
  VADD.F32 S9, S18  ;Se suma el resultado actual al resultado de la iteracion actual
  VLDR.F32 S2, =1   ;Se reinicia el valor de S2 a 1
  VLDR.F32 S16, =1  ;Se reinicia el valor de S16 a 1
  VLDR.F32 S11, =1  ;Se reinicia el valor de S11 a 1
  VCMP.F32 S4, S6   ;Se compara si S4(n)=S6(iteraciones)
  VMRS APSR_nzcv, FPSCR ;Realizamos un traslado de banderas
  BNE sumatoria ;Se reinicia la funcion si no se ha cumplido la condicion de iteraciones
  VADD.F32 S9, S0 ;Se suman el resultado en S9 al valor de x     VADD.F32 S9, S0 ---------->>>>>>>>>PARA COSENO SUSTITUIR POR "VADD.F32 S9, S16"
  ;VADD.F32 S9, S16 ;<<<-----COSENO
Loop
  B Loop

factorial   ;Determina el factorial del denominador, S2 almacena el producto, S8 decrece hasta llegar a
  VMUL.F32 S2, S8 ;Multiplicamos S2 por el factorial
  VSUB.F32 S8, S7 ;Decrecemos el registro S8 en 1
  VCMP.F32 S8, S7 ;Comparamos si el registro S8 ya ha llegado a 1
  VMRS APSR_nzcv, FPSCR ;Realizamos un traslado de banderas
  BNE factorial ;Si la condicion de S8=S7 no se cumple se repite la funcion factorial
  BX LR ;Se retorna a la funcion principal sumatoria

potenciaa ;Representa la potencia del numerador
```

```
    (-1)^n
  VMUL.F32 S11, S5  ;Se multiplica S11(1) por S5 (-1)
  VSUB.F32 S12, S7  ;Se decrece S12 en 1
  VCMP.F32 S12, S13 ;Se compara S12 con S13(0)
  VMRS APSR_nzcv, FPSCR ;Realizamos un traslado de banderas
  BNE potenciaa ;Si la condicion S12=13 no se cumple se repite la funcion
  BX LR ;Se retorna a la funcion principal

potenciab ;Representa la potencia (x)^(2n+1)
  VMUL.F32 S16, S0  ;Se multiplica el valor de x por su anterior valor
  VSUB.F32 S15, S7  ;Se decrece el valor de S15 (2n+1) en 1
  VCMP.F32 S15, S13 ;Se compara si S15 ya ha llegado a S13(0)
  VMRS APSR_nzcv, FPSCR ;Realizamos un traslado de banderas
  BNE potenciab ;Si la condicion S15=S13 no se cumple se repite la funcion
  BX LR

  ALIGN
  END
```

Listing 1: Entorno de keil uvision

VI. RESULTADOS

A. Demostración del respectivo código

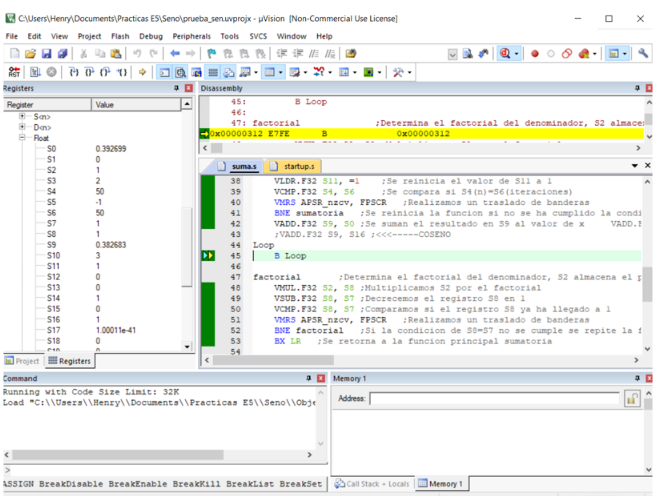

Figura 4: Resultado final del seno del respectivo ángulo

Resultado final del programa	**Resultado final calculadora**
0.382683	0.3826834324

Fuente: Elaboración propia 2024

VII. DISCUSIÓN DE RESULTADOS

El código demuestra una implementación robusta y eficiente de la serie de Taylor para el cálculo de la función seno en lenguaje ensamblador. Si bien cumple su propósito de manera efectiva, hay áreas donde podría ser mejorado para aumentar su precisión y flexibilidad. Este código es un ejemplo excelente de cómo aplicar técnicas matemáticas avanzadas en un contexto de programación de bajo nivel, y proporciona una base sólida para desarrollos futuros en aplicaciones similares.

VIII. CONCLUSIONES

- Cálculo Eficiente de Series de Taylor: El programa demuestra cómo se puede utilizar la serie de Taylor para aproximar funciones trigonométricas, en este caso, el seno de un ángulo dado en radianes. La serie de Taylor es una herramienta matemática poderosa para aproximar funciones mediante sumas de términos polinomiales, y este programa ilustra cómo implementarla en lenguaje ensamblador.
- Importancia de la Precisión y Control en Ensamblador: El programa subraya la importancia de la precisión en el control del flujo de ejecución y el manejo de los registros en lenguaje ensamblador. Pequeños errores en la lógica o en la manipulación de registros podrían llevar a resultados incorrectos, lo que resalta la necesidad de un cuidado especial al programar a este nivel.
- Uso de Bucle Infinito para Mantener Resultados: El bucle infinito al final del programa es una práctica común en aplicaciones embebidas donde se requiere que el programa continúe corriendo indefinidamente, manteniendo los resultados disponibles en los registros para futuras referencias o procesos.

IX. REPOSITORIO

`https://github.com/Henry-Chacon/Practica_1_LAB_E5`. [Fuente: Elaboración propia 2024].