Laboratorio 2. Acercándose al Hardware: Programación en Lenguaje Ensamblador

Objetivos de aprendizaje

- Usar MARS (un *IDE* para MIPS) para escribir, ensamblar y depurar programas MIPS
- Escribir programas MIPS incluyendo instrucciones aritméticas, de salto y memoria
- Comprender el uso de subrutinas en MIPS, incluyendo el manejo del *stack*
- Realizar llamadas de sistema en MIPS mediante "*syscall*"
- Implementar algoritmos en MIPS para resolver problemas matemáticos sencillos

Entrega

Suba los archivos creados, junto con su informe, a través del UVirtual del curso. Todos los archivos con código MIPS deben poder ensamblarse y ejecutarse en el simulador MARS, y deben estar debidamente comentados. Guarde cada programa en un archivo distinto con el nombre de la parte que corresponde. Por ejemplo, el trabajo de la Parte 1, pregunta A, guárdelo en el archivo "parte1a.asm".

Parte 1: Uso de syscall

Escriba un programa que lea dos enteros ingresados por el usuario, determine el máximo entre estos dos números y luego imprima este valor. Para ello, utilice llamadas de sistema, "syscall", para 1) imprimir en la consola de MARS los mensajes de interacción con el usuario, 2) permitir que el usuario ingrese los números en tiempo de ejecución, 3) imprimir el resultado en la consola, y 4) terminar el programa (*exit*). (Note que en la ayuda de MARS puede encontrar la documentación sobre el uso de *syscall*). El cálculo del máximo debe ser realizado en una subrutina, utilizando los registros apropiados para argumentos y salida de un procedimiento. La consola de entrada/salida de MARS debe lucir así (los números son ejemplos):

```
Por favor ingrese el primer entero:  31
Por favor ingrese el segundo entero: 10
El maximo es: 31
-- program is finished running --
```

Parte 2: Subrutinas para encontrar el máximo común divisor

Escriba un programa en MIPS que calcule el máximo común divisor entre dos números enteros mediante la implementación de subrutinas. Para ello, implemente una forma recursiva del algoritmo de Euclides. Para este programa, los operandos deben estar escritos "en duro" en el mismo código (*i.e.*, no es necesario pedirlos al usuario vía consola) y deben estar claramente identificados para poder probar con otros valores al evaluar el trabajo.

Parte 3: Subrutinas para multiplicación y división de números

A) Escriba un programa en MIPS que calcule la multiplicación de dos enteros mediante la implementación de subrutinas. **No** se pueden utilizar instrucciones de multiplicación, división y desplazamiento: mul, mul.d, mul.s, mulo, mulou, mult, multu, mulu, div, divu, rem, sll, sllv, sra, srav, srl, srlv; sino que se debe implementar una técnica de multiplicación basada en otras operaciones matemáticas y el uso de subrutinas.

B) Escribe un programa en MIPS similar al de A) que calcule la división de dos enteros mediante la implementación de subrutinas. Al igual que en A), **no** se pueden utilizar instrucciones de multiplicación, división y desplazamiento: mul, mul.d, mul.s, mulo, mulou, mult, multu, mulu, div, divu, rem, sll, sllv, sra, srav, srl, srlv; sino que se debe implementar una técnica de división basada en otras operaciones matemáticas y el uso de subrutinas.

Parte 4: Subrutinas para multiplicación y división de números

A) Escriba un programa que calcule una aproximación de las siguientes funciones, evaluadas en un número cercano a cero:
   1) Seno
   2) Logaritmo natural

   Para aproximar estas funciones, utilice expansiones de Taylor en torno a 0. El programa **debe** utilizar para el cálculo de las multiplicaciones y divisiones los procedimientos de la Parte 3. Además, el programa debe pedir al usuario, vía consola MARS, el número para el cual se quieren evaluar las funciones, y debe mostrar en la misma consola los resultados.

B) Cuantifique el error de la aproximación de la parte A) en función del número de términos de la expansión de Taylor. Para ello, construya un gráfico que muestre el error para 3 números distintos utilizando la expansión de orden 1 hasta, al menos, orden 11.

Informe

El informe para entregar debe contar con lo siguiente:

- Introducción que incluya el problema, solución y objetivos de esta experiencia
- Marco teórico que explique los conceptos necesarios para entender el trabajo desarrollado
- Explicación breve del desarrollo de la solución y cómo se llegó a esta
- Resultados de cada parte del laboratorio
- Conclusiones

Exigencias

- El informe escrito debe ser entregado en formato PDF y no puede exceder 10 páginas de contenido, sin considerar portada ni índice. En caso contrario, por cada página extra se descontará 5 décimas a la nota final.

- Tanto el código fuente como el informe deben ser enviados por medio de la plataforma UVirtual en un archivo comprimido, cuyo nombre debe incluir el RUT de el/la alumno/a (ej.: lab2_12345678-9.zip).

Recomendaciones

- **Consultar al ayudante en las sesiones de laboratorio.**
- En caso de dudas o problemas en el desarrollo, asista a la sesión de laboratorio.

Descuentos

- Por cada día de atraso se descontará un punto a la nota final de este laboratorio.
- Por cada tres faltas ortográficas o gramaticales en el informe, se descontará una décima a la nota del informe.
- Por cada falta de formato en el informe se descontará una décima a la nota del informe.
- Por cada página extra en el informe se descontarán 5 décimas a la nota final del informe.

Evaluación

- La nota del laboratorio será el promedio aritmético del código fuente con el informe.
- En caso de que no se entregue el informe o el código fuente, se evaluará con la nota mínima.
- Este laboratorio debe ser entregado el viernes 29 de noviembre de 2021, hasta las 23:59 hrs.