Facultad de Informática
Universidad Complutense de Madrid

Fundamentos de Computadores II - Práctica 5
Desarrollo en ensamblador

Daniel Báscones (danibasc@ucm.es)

25 de abril de 2023

1. Objetivos

En esta práctica aplicaremos todas las técnicas aprendidas en ensamblador para crear nuestro propio programa. Los objetivos son:

- Demostrar la capacidad de desarrollar un programa en ensamblador.
- Demostrar habilidad a la hora de dividir un programa en diferentes archivos y funciones.
- Comprender cómo escribir código ensamblador para que cumpla los estándares.

2. Desarrollo de la práctica

2.1. Requisitos obligatorios

El programa creado deberá cumplir los siguientes requisitos, **todos ellos en ensamblador**:

1. El programa debe tener una función `main`, que sea el punto de entrada del programa.
2. Las variables de entrada y salida del programa deben estar en ensamblador, en las secciones adecuadas a tal efecto.
3. El programa debe, en algún momento, utilizar condicionales (`if`).
4. El programa debe, en algún momento, usar bucles en ensamblador (`for/while`) para recorrer arrays y realizar algún cómputo con/sobre sus valores.
5. Debe haber dos funciones adicionales a `main` en ensamblador:

   a) Una función hoja.

   b) Una función no hoja que llame a la primera.

6. El programa debe, en algún momento, pasar como argumentos a función tanto valores escalares, como arrays.

7. El código debe estar organizado en al menos dos archivos de código fuente.

Es obligatorio documentar bien el código desarrollado, con comentarios tanto en ensamblador como en C si es que se incluyese. Un buen desarrollo no incluye solo un código funcional, sino un código comprensible.

Se pedirá una explicación de qué hace el código y cómo lo hace, y debe tener una aplicabilidad en el mundo real. (Ejemplo: ordenación, búsqueda, cálculos vectoriales, cálculos matriciales, accesos a estructuras, u otros algoritmos que hagan operaciones en un array. Ante la duda, consulta con tu profesor).

El código desarrollado debe ser original y no provenir de las prácticas anteriores. Sí puede estar basado en código C ya existente, que habrá que traducir. **Nota:** Al tomar un código C como referencia, aseguraos de quitar llamadas a funciones que no hemos utilizado en ensamblador (como por ejemplo `printf/cin/cout`) para evitar problemas en la compilación. El código de referencia debe ser un C puro sin llamadas a librerías.

2.2. Requisitos opcionales

Opcionalmente, se puede añadir todo el código C que se desee alrededor del ensamblador requerido. Una opción válida es escoger algún algoritmo ya implementado en C, y trasladar parte o su totalidad a ensamblador. Nótese que el punto de entrada al programa debe ser ensamblador, así como las funciones hoja y no hoja especificadas en el apartado anterior.

2.3. Evaluación

Se deberá demostrar conocimiento sobre el programa desarrollado, y capacidad para mostrar los resultados en registros y memoria dentro del entorno del simulador.

Se deberá ser capaz de probar el correcto funcionamiento del programa bajo varios casos de prueba, no un sólo caso sintético. Se pedirán cambios en la entrada a la hora de evaluar.

3. Ejemplo

Por ejemplo, se puede escoger uno de los algoritmos de ordenación de la colección de **TheAlgorithms** (`https://github.com/TheAlgorithms/C/tree/master/sorting`), creando un punto de entrada en ensamblador, y traduciendo un par de funciones (hoja y no hoja) de las ofrecidas en uno de los ejemplos a ensamblador. El resto del código puede mantenerse en C.