**PRÁCTICA 1**

Desarrollo de código para el procesador ARM

DANIEL HUICI

ÍNDICE

[RESUMEN 3](#_Toc22605457)

[METODOLOGÍA 4](#_Toc22605458)

[MARCOS DE PILA 4](#_Toc22605459)

# RESUMEN

A lo largo de esta práctica 1, se ha ido desarrollando una serie de objetivos propuestos con el fin de obtener conocimientos sobre el funcionamiento de la placa Embest S3CEV40, y la generación y optimización de ciertas funciones en código ensamblador, pudiendo comprobar además las optimizaciones gracias a los “timers” desarrollados en esta misma práctica.

Durante el transcurso de esta práctica, se han consultado toda la documentación necesaria referente al desarrollo sobre la placa, así como documentación referente a la gestión del procesador ARM. Se ha realizado también un estudio en profundidad sobre el juego propuesto “reversi”, el cuál ha sido protagonista de gran parte de los retos de esta práctica.

Siguiendo el procedimiento de la misma, se han realizado dos códigos ensambladores en C, que imitan el funcionamiento de una de las funciones dentro del juego propuesto, “patron\_volteo”: una primera versión en la que, dentro del propio código se hace una llamada externa a una función en C (“ficha\_valida”), y otra en la que dicha función es incluida dentro del propio código ensamblador. En ambas versiones del código se garantiza el seguimiento del estándar AACPS.

Con el fin de comprobar el correcto funcionamiento de los códigos anteriormente mencionados, se ha añadido código extra al juego propuesto “reversi”. Se comprueba que se realicen correctamente una serie de movimientos en determinadas direcciones.

Por último, se ha desarrollado el código necesario para hacer funcionar uno de los timers que proporciona la placa y poder tomar medidas de tiempo sobre los códigos anteriormente desarrollados, teniendo en cuenta todos los aspectos posibles para evitar errores en las medidas.

A lo largo de esta memoria, se desarrollará cada uno de los puntos anteriormente mencionados para lograr que se entienda de la forma más exacta y concisa posible todos los pasos realizados, y el porqué de cada uno.

# METODOLOGÍA

## MARCOS DE PILA

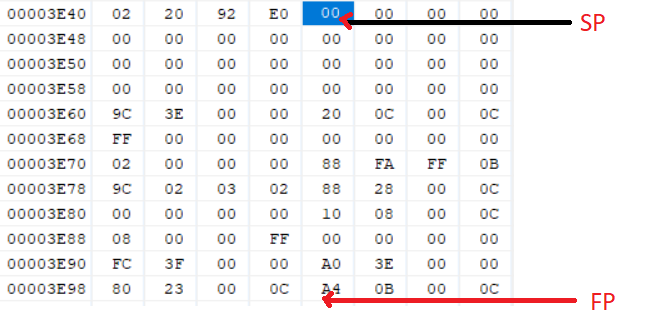
Con el objetivo de realizar correctamente los códigos en ensamblador, ha sido de vital importancia una buena gestión de los marcos de pila.

Al estar desarrollando un código en ensamblador, es nuestra competencia que en la pila se apilen y se desapilen los registros correctos, en vez de ser el compilador quién se encarga normalmente, cuando trabajamos directamente en alto nivel.

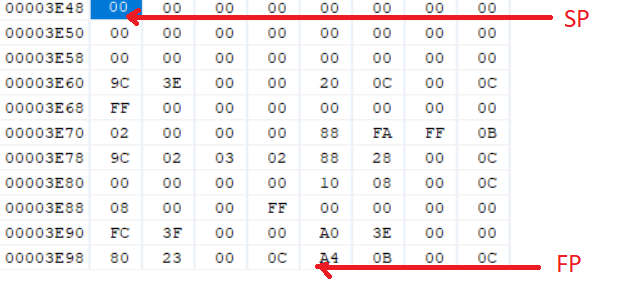
Siguiendo el estándar AACPS, como máximo usaremos 4 registros para el paso de parámetros en las llamadas a subrutinas (registros r0-r3). El resto de parámetros deberán ir apilados en la pila, y habrá que hacer accesos a memoria para obtenerlos dentro de la subrutina

Para este caso hacemos usos de los registros del r4 al r10 (r4 a r9 para la versión arm\_arm). Desconocemos el uso que está dando el compilador a estos registros en anteriores subrutinas, y podríamos generar errores en los valores si no guardamos en la pila. Es por esto que, antes de nada, apilamos los registros en la pila con una instrucción push. Tampoco debemos olvidarnos de guardar el registro fp, y el registro lr, para saber a qué dirección hay que retornar una vez finalizada la subrutina en cuestión.

Una vez completada la subrutina, simplemente se desapilan los valores anteriormente apilados para recuperar el contexto



Marco de pila patron\_volteo\_arm\_c



Marco de pila patron\_volteo\_arm\_arm