Resumen Práctica 1

Durante la práctica se muestra el proceso de optimización del juego **Reversi**. A partir de un proyecto proporcionado en lenguaje c se crean dos funciones diferentes, con el objetivo de sustituir a la función más costosa. Ambas se crean para intentar reducir costes en espacio y tiempo. Para su implementación se emplean técnicas diferentes.

La primera de ellas, `patron_volteo_arm_c` consiste en la función `patron_volteo` pero escrita en ensamblador ARM y conservando las llamadas a la subrutina `ficha_valida` en c. Para optimizarla se ha hecho uso de instrucciones condicionales, acciones múltiples, además de la reducción de lecturas/escrituras entre otras muchas cosas. Pero sobre todo se han centrado los esfuerzos en intentar reducir el tamaño de su bloque de activación, el cual se trata de una de las partes más costosas del juego. Para ello se deja de apilar el *link register*.

La segunda, `patron_volteo_arm_arm`, también sustituye a `patron_volteo`. Al igual que `patron_volteo_arm_c` está escrita en ARM, pero sin embargo no incluye las llamadas a `ficha_valida` porque hace uso de la técnica *inlining*, que consiste en incrustar el código de la propia función sustituyendo a su llamada. Con esto se consigue un ahorro bastante significativo en tiempo. De nuevo se procura también reducir el tamaño de la pila, consiguiendo esta vez no tener que apilar *fp* ni *lr*.

Finalmente, se hace un análisis en el que se compara para un mismo movimiento el rendimiento en tiempo y tamaño de las distintas versiones del código llegando a la conclusión de que la mejor es la ARM usando *inlining*.