Tarea optimización con CCS

El fin del trabajo es la aplicación de las diferentes técnicas para optimizar DSP vistas en clase, usando el entorno Code Composer. Cuenta con dos partes:

1) La implementación y optimización de un filtro FIR.
2) La optimización de un código previamente dado.

Como recordatorio, el filtro FIR consiste en un algoritmo del tipo:

$$Y(i) = \sum_{j=0}^{N-1} c(k) * x(i-j)$$

- Donde i se encuentra entre 0 y M.
- M es el número de puntos de la señal.
- N es el número de términos no nulo, y de coeficientes del filtro.

Tareas a realizar

Implementación de código

1. Implementar la función del Filtro FIR para que se pueda trabajar con un vector de datos y un vector de coeficientes cuyo tamaño se pueda modificar fácilmente. Ambos vectores son pasados a la función como argumentos. Dicha función tiene ser invocada desde el main y retornar un vector.

   Se puede realizar un programa de prueba con pocos datos para comprobar la corrección del algoritmo pero, para esta parte inicial de la práctica, el tamaño mínimo debe ser de **1000 valores el vector datos** y **15 el vector coeficientes**. Los valores iniciales pueden ser definidos al azar.

2. La segunda versión del programa efectúa la lectura de los valores de los coeficientes (archivo coeficientes.csv) y de los valores del archivo musica4.csv. Son 21 coeficientes. Del archivo musica4.csv no hace falta obtener todos los valores, porque hablamos de un fichero muy grande. Se puede jugar con el tamaño de los datos en las pruebas. El mínimo será 2000. Además, los vectores hay que pasárselos como punteros. En este caso no retorna el vector, sino el vector resultante también es pasado como un puntero a la función. De forma similar a la que hemos trabajado en clase.

Optimización

La optimización se realiza a partir de la segunda versión del código previamente realizado, y del programa suministrado, llamado *practica.c*. Haz uso del depurador para asegurarte de que los resultados sigan dando igual después de la optimización. Por otro lado, se debe llevar a cabo el profiling del código para obtener información acerca de los ciclos de CPU que tarda en ejecutarse. Se puede utilizar cualquiera de las opciones de profiling vistas en clase (indica en el informe cuál usaste). Se tienen que realizar diversas versiones del código, para posteriormente compararlas. Toda versión incluye los cambios hechos previamente.

1. La **primera versión** del código se corresponde con la *practica.c* sin modificar, y a la segunda implementación del filtro FIR.
2. La **segunda versión** debe incluir el uso de keywords, como Const y Restrict, en las variables que consideres.
3. La **tercera versión** incluye el desenrollado manual y la optimización de bucles, condicionales, tipos y funciones, con las técnicas vistas en clase. Lo que también implica el desenrollado del prólogo y epílogo del filtro FIR.
4. La **cuarta versión** del código implica la utilización de pragmas. Pudiendo llegar a sustituir el desenrollado manual hecho por los pragmas, en la situaciones donde lo consideres.
5. La **quinta versión**, y última, tienes que incluir intrínsecos. De igual manera que en el desarrollo de la versión previa, puedes tanto añadir intrínsecos como sustituir código ya realizado por estos.

Tienes que aplicar los diferentes niveles de optimización y apuntarlos, para cada una de las 5 versiones, con el fin de compararlos. Eso quiere decir con la optimización apagada, y con los niveles 0, 1, 2 y 3.

*La investigación autónoma y uso de intrínsics y/o pragmas no utilizados en clase se valorará positivamente.
** Lo que tienes que medir es lo que tarda la función **filtroFir**, y la función **practica**, no los bloques u otras funciones que incluyan. Para comprobar de manera correcta el tiempo, borra/comenta los prints internos colocados para la depuración del código.

Documentación a entregar

Se generará un fichero ZIP con la siguiente denominación de NOMBRE_FIR_2022 donde NOMBRE representa el nombre y primer apellido del alumno para poder identificarlo. El fichero debe contener:

- Los proyectos sin ejecutables para poder ser compilados por el profesor con las diferentes versiones del código.
- Un informe, con una buena estructura, es decir, introducción, desarrollo y conclusión, que indique:
    - Características del proyecto.

    - Explicación del código C. Tanto el proceso del desarrollo del código del filtro Fir, como los procesos para optimizarlos. Asegúrense de que el resultado obtenido por las funciones es siempre el mismo, independientemente de las técnicas de optimización aplicadas.
    - Tablas comparativas (también se puede complementar con gráficas) con los ciclos de reloj para cada una de las versiones llevadas a cabo.
    - Valoración personal. Es decir, deducir por qué se mejora, o no, el número de ejecuciones en ciclos de CPU. La finalidad no es lograr el código que se ejecute en menos ciclos de reloj, sino entender y explicar qué sucede con cada mejora y nivel de optimización. Fundamentándolo.

Puntuación

- Implementación del código del filtro FIR. **(2 puntos) (Necesario para corregir todo lo demás)**
    - Implementación Filtro FIR para un vector de 1000 valores y otro de 15 coeficientes. **(0.5 puntos)**
    - Implementación Filtro FIR con el uso de los ficheros indicados para leer coeficientes y valores. **(1 puntos)**
- **Segunda versión** del código con keywords. **(0.3 puntos)**
- **Tercera versión** del código con desenrollo y otras técnicas para bucles y condicionales. **(1.2 puntos)**
- **Cuarta versión** del código con pragmas. **(0.9 puntos)**
- **Quinta versión** del código con intrínsecos. **(1.2 puntos)**
- Calidad y limpieza de código. **(0.9 puntos)**
- Informe realizado. **(4 puntos)**
    - Estructura informe. **(1.2 puntos)**
    - Profiling y valoración personal. **(2 puntos)**
    - Investigación y explicación de los intrínsecos y pragmas utilizados. **(0.6 puntos)**
    - Coherencia y calidad del texto e imágenes del informe. **(1.2 puntos)**