INTEGRANTES DEL GRUPO:

- David Gabriel Suárez (uo270943) - Moisés Sanjurjo Sánchez (uo270824) - Adolfo Rodríguez Sánchez(uo271620) - Federico Cuervo Ricardo (uo67609)

Partiendo de la Fase 1 del trabajo entregada anteriormente, procedimos a plantear las modificaciones necesarias para incorporar al código el uso de las instrucciones de bajo nivel SIMD. En el caso de nuestro grupo, al que se facilitó un tamaño de paquete de datos de 256 bits y un tipo de datos flotante de precisión sencilla o 32 bits, nos ceñimos al uso del vector de empaquetado de datos m256, o vector de 256 bits, que almacena hasta 8 datos de tipo flotante de simple precisión, para operar con ellos simultáneamente. La máquina en la que evaluamos el rendimiento de ejecución del algoritmo soporta como juego de instrucciones SIMD más avanzado el AVX, por lo que las instrucciones a utilizar en nuestra implementación son esas.

A modo de descripción general, los algoritmos que hemos implementado cargan dos imágenes fuente mediante la biblioteca CImage, al igual que el programa mono hilo. Para el tipo de datos flotante de 32 bits que se nos suministró, calculamos el número de datos que nos caben en el paquete de datos de 256 bits que también nos fijaba la especificación, dando como resultado vectores de 8 números flotantes de simple precisión o 32 bits, que serán operados de manera simultánea en cada instrucción aritmética que se ejecute dentro del algoritmo de procesado de las imágenes. Empleando un bucle for que avanzará de 8 en 8 posiciones por los pixeles de las imágenes fuente, hasta alcanzar los totales disponibles, procedemos a la carga de los mismos en un array de memoria alineada para cada una de las dos imágenes (que al igual que en el programa mono hilo, han de ser de la misma anchura y altura, además de formato). Una vez cargados esos 8 pixeles, se multiplican para cada imagen por su valor de proporción para su participación en la imagen final, se suman los resultados de dichos productos y, por último, antes de almacenarlos en la zona de memoria que se empleará para persistir la imagen final, se realiza una operación de acotado, también mediante función intrínseca, de dicho resultado, para asegurar que su valor no sobrepase el límite válido de 255.

A continuación, se resumen las mediciones de tiempos de ejecución en condiciones comparables a las del algoritmo implementado en la fase 1:

	Media ts1 a ts10 (s)	Productividad (tareas/min)	Aceleración
SingleThread	7,92	7,58	-
SingleThread-SIMD	4,68	12,82	1,69

Para un examen más detallado de las mediciones, por favor acuda a la hoja Excel adjunta al proyecto.

La transformación inicial del algoritmo desde la versión mono hilo hasta la versión SIMD implementada mediante funciones intrínsecas fue realizada por Federico Cuervo Ricardo, mientras que en posteriores reuniones, extensas, se detectaron ineficiencias y código redundante e innecesario, que fue eliminado para reducir la complejidad del programa, por lo que la participación del equipo al completo en la elaboración de la implementación final ha sido bastante equilibrada.

- Federico Cuervo Ricardo: Transformación inicial del algoritmo mono hilo en una versión basada en instrucciones SIMD. Elaboración del esqueleto de la primera versión de la memoria.
- David Gabriel Suárez: Participación en las reuniones para depurar y optimizar algoritmo inicial y eliminar código redundante.
- Adolfo Rodríguez Sánchez: Participación en las reuniones de depuración y optimización del código y finalización de la memoria, junto con las mediciones de los tiempos en la máquina de pruebas y elaboración del informe sobre los mismos.
- Moisés Sanjurjo Sánchez: Participación en las reuniones de depuración y optimización del algoritmo inicial y búsqueda de alternativas a partes del algoritmo que no funcionaban correctamente en la versión inicial.