Uso de ChatGPT e IA Generativa:

En este práctico se permite el uso de estas herramientas para:

- **Generar el código del ejercicio 1.**
- **Generar gráficas y tablas a partir de los resultados, corregir ortografía y redacción, etc.**
- **Generar scripts que automaticen las pruebas a realizar.**

NO se permite el uso de estas herramientas para:

- **Generar el código del ejercicio 2.**
- **Generar una interpretación de los resultados.**

Práctico 1 - Patrones de acceso a memoria

El objetivo de este práctico es reflexionar sobre la jerarquía de memoria, en especial la memoria principal y los distintos niveles de caché, y sobre cómo distintos patrones de acceso a los datos hacen un uso distinto de dicha jerarquía. En los siguientes ejercicios se accederá a una estructura de datos realizando la misma cantidad de operaciones, aunque en distinto orden. Para que los tiempos de ejecución sean lo más estables posible se recomienda ejecutar en un sistema con poca carga, medir tiempos lo suficientemente grandes (adaptando el tamaño de la entrada o la cantidad de repeticiones de las pruebas), y evitar la utilización de máquinas virtuales.

Ejercicio 1	Localidad espacial

1. Escriba un programa en C/C++ que reserve e inicialice un arreglo de `char` de gran tamaño (por ejemplo 100MB). A continuación el programa debe recorrer el arreglo (por ejemplo incrementando el valor de cada posición) de manera secuencial (primero la posición 0, luego la 1, y así sucesivamente). Durante la recorrida, el siguiente índice a visitar debe leerse de un arreglo inicializado previamente. Registre el tiempo de ejecución de la recorrida.

2. Realice otra recorrida por el arreglo que visite la misma cantidad de elementos pero realizando saltos aleatorios. Durante la recorrida, el siguiente índice a visitar debe leerse de un arreglo inicializado previamente. Mida el tiempo de ejecución y reflexione sobre los resultados.

Ejercicio 2	Localidad temporal

Considere la siguiente función para multiplicar una matriz A por un vector B de gran tamaño:

```
void matrix_mult(float **A, float *B, float *C, size_t size) {

    for (size_t i = 0; i < size; i++) C[i] = 0;

    for (size_t i = 0; i < size; i++)
        for (size_t j = 0; j < size; j++)
            C[i] += A[i][j] * B[j];
}
```

Tal como está el código, por cada fila de A se recorre todo el vector B. Si la matriz es suficientemente grande, al pasar a la siguiente fila de A, los primeros elementos de B ya no se encuentran en la caché. Note que cada elemento de A se accede solamente 1 vez (no es posible explotar la localidad temporal) pero cada elemento de B se accede `size` veces.

Si se modifica el código de la siguiente forma:

```
void matrix_mult(float **A, float *B, float *C, size_t size) {

    for (size_t i = 0; i < size; i++) C[i] = 0;

    for (size_t jj = 0; jj < size; jj+=TAM_BL)
        for (size_t i = 0; i < size; i++)
            for (size_t j = jj; j < jj+TAM_BL; j++)
                C[i] += A[i][j] * B[j];
}
```

se realizan todos los productos correspondientes a un pequeño bloque de B (de tamaño `TAM_BL`) antes de pasar al siguiente bloque, por lo que al cambiar de fila en A los elementos de B no deberían haber sido reemplazados en la caché.

1. Escriba una función para el producto de matrices extendiendo la función anterior (ahora B y C tendrán `size` columnas). Mida el desempeño en MFLOPS[1] para valores de `size` de distinto orden (comparables al tamaño de los distintos niveles caché).

2. Aplique la técnica descrita para el producto matriz-vector al producto de matrices, explotando la localidad temporal tanto en A como en B. Compare los tiempos de ejecución y desempeño en MFLOPS de ambas versiones para distintos tamaños de matriz y bloque. Reflexione acerca de los resultados.

Entrega

- Se debe entregar un informe en PDF con la solución de los ejercicios que contenga, **como máximo, 4 páginas**.
- No se entrega el código fuente.
- Las secciones relevantes del código deben figurar en el informe.

[1] $\texttt{size}^3/(\text{segundos}\times 10^6)$