EJERCICIO 1

Nombre: Javier Menacho Paca **CI**: 12763905
Docente: Ph. D. Moises Martin Silva Choque **Materia**: INF-317

Explique cuál es la relación de la taxonomía de Flynn y cada una de las librerías utilizadas hasta el momento.

En primera instancia se definirá la taxonomía de Flynn:

La taxonomía de Flynn es un concepto utilizado en el ámbito de la informática y la arquitectura de computadoras para clasificar los modelos de computación paralela y sistemas de computación según el número de flujos de instrucciones y datos que pueden procesar simultáneamente. Fue propuesta por Michael J. Flynn en 1966 y se ha convertido en un marco de referencia importante para comprender la clasificación de arquitecturas de computadoras en términos de paralelismo.

Taxonomía de Flynn

	Una instrucción	Múltiples instrucciones
Un dato	SISD	MISD
Múltiples datos	SIMD	MIMD

La taxonomía de Flynn se divide en cuatro categorías principales, que son las siguientes:

SISD (Single Instruction, Single Data): En esta categoría, una sola instrucción se ejecuta en una sola unidad de procesamiento, que opera en un solo conjunto de datos a la vez. Esto es esencialmente la arquitectura de computadora convencional, donde una sola CPU ejecuta una secuencia de instrucciones en un solo conjunto de datos.

SIMD (Single Instruction, Multiple Data): En esta categoría, una sola instrucción se ejecuta en múltiples unidades de procesamiento, pero todas las unidades de procesamiento operan en su propio conjunto de datos. Esto es común en procesadores vectoriales y en aplicaciones que requieren realizar la misma operación en múltiples datos, como gráficos o procesamiento de señales.

MISD (Multiple Instruction, Single Data): En esta categoría, múltiples instrucciones se ejecutan en una sola unidad de procesamiento, que opera en un solo conjunto de datos. Aunque esta categoría es rara y menos utilizada en comparación con las otras, algunos ejemplos teóricos y experimentales han sido propuestos, como en sistemas de control y redundancia.

MIMD (Multiple Instruction, Multiple Data): En esta categoría, múltiples instrucciones se ejecutan en múltiples unidades de procesamiento, y cada unidad de procesamiento puede operar en su propio conjunto de datos. Este es el modelo más común en sistemas de cómputo paralelo, como clústeres de computadoras, sistemas multiprocesador y supercomputadoras, donde múltiples hilos o procesadores trabajan de manera independiente en sus propias tareas y datos.

RELACIÓN CON LAS LIBRERÍAS UTILIZADAS HASTA EL MOMENTO:

OpenMP (C) y MIMD: OpenMP es una API que se utiliza en sistemas MIMD para aprovechar el paralelismo en programas. Permite crear múltiples hilos de ejecución (flujos de instrucciones) que trabajan en paralelo y pueden procesar datos de manera simultánea. Estos hilos pueden ejecutarse en núcleos de CPU independientes, lo que es coherente con la arquitectura MIMD.

MPI (C) y MIMD: MPI es una biblioteca de comunicación diseñada para sistemas MIMD distribuidos o paralelos. Permite a múltiples procesos (flujos de instrucciones) ejecutarse en paralelo en varios nodos de un clúster o sistema distribuido. Los procesos pueden comunicarse entre sí enviando mensajes (datos) a través de la red y sincronizarse para coordinar tareas paralelas.

MULTIPROCESSING (PYTHON) CON MIMD Y SISD:

- Implica que en un sistema MIMD, múltiples flujos de instrucciones pueden ejecutarse de manera independiente en múltiples flujos de datos, lo que es consistente con la forma en que multiprocessing permite la programación paralela en Python.
- Si bien multiprocessing se utiliza principalmente para ejecutar múltiples flujos de instrucciones en paralelo, no se limita exclusivamente a sistemas MIMD. En algunos casos, podría utilizarse para crear múltiples procesos que realizan la misma tarea en diferentes conjuntos de datos con SISD.

THREADING (C#) Y MIMD: La relación entre el threading en C# y la taxonomía de Flynn se encuentra en que, en C#, puedes utilizar múltiples hilos de ejecución para lograr paralelismo, lo que se asemeja a la categoría MIMD de la taxonomía de Flynn. Cada hilo en C# puede ejecutar instrucciones de manera independiente y trabajar con diferentes conjuntos de datos. Esto permite aprovechar el paralelismo en tareas como cálculos intensivos, operaciones de E/S, procesamiento de datos, etc.