Examen práctico INF 317

Nombre: Basilio Ticona Pucho 9887388

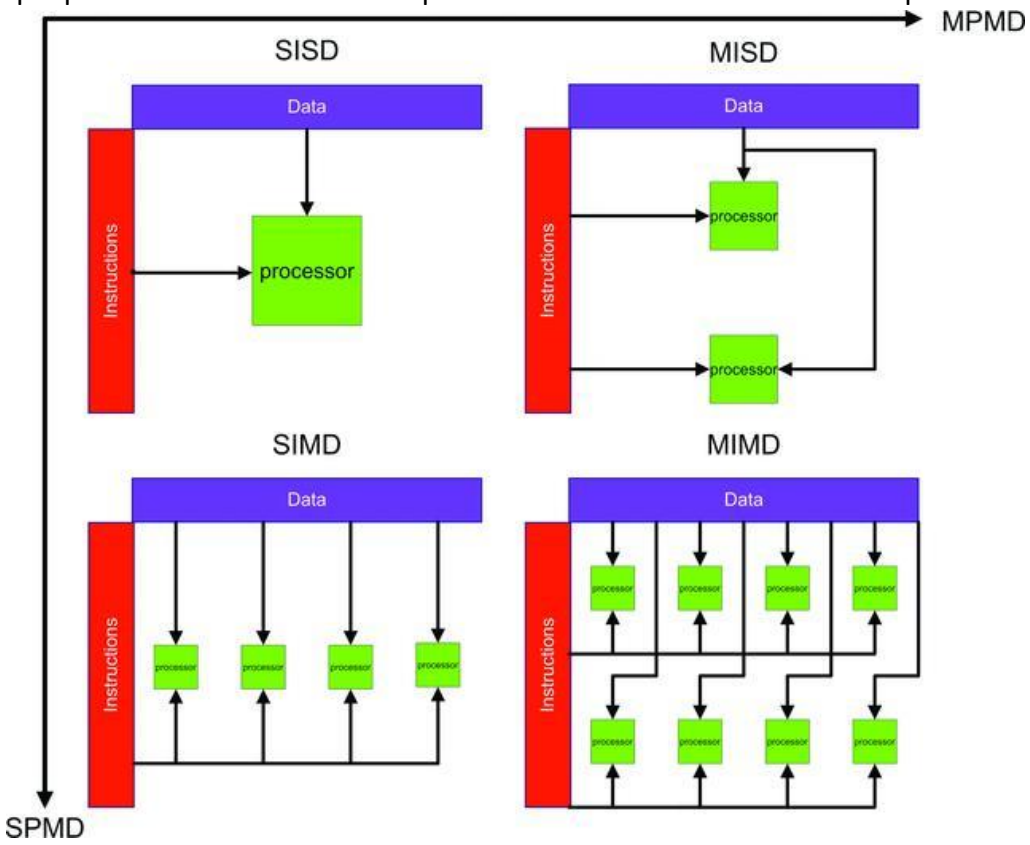
1. **Describa cada una de las siguientes: SISD, SIMD, MISD y MIMD**

   SISD (Single Instruction Single Data): Esta es la arquitectura de computadora más básica y sencilla, en la que se ejecuta una única instrucción en una sola unidad de datos. Esto significa que solo se procesa una instrucción en un momento dado y se trabaja con un solo flujo de datos. Es el método clásico de procesamiento secuencial.

   SIMD (Single Instruction Multiple Data): En esta arquitectura, se ejecuta una única instrucción en múltiples unidades de datos al mismo tiempo. Para lograr esto, se utilizan varios procesadores que trabajan en paralelo en un conjunto de datos idénticos. En otras palabras, se procesa una sola instrucción en diferentes conjuntos de datos.

   MISD (Multiple Instruction Single Data): Esta arquitectura es poco frecuente en la actualidad, en ella se ejecutan múltiples instrucciones en una sola unidad de datos. Es decir, varias instrucciones se procesan en paralelo sobre un único flujo de datos.

   MIMD (Multiple Instruction Multiple Data): Esta arquitectura es la más complicada porque varias instrucciones se ejecutan en múltiples unidades de datos simultáneamente. Cada unidad de procesamiento opera de manera autónoma, lo que permite realizar tareas completamente diferentes al mismo tiempo.



2. **De la anterior describa cual se aplica en: OpenMP, MPI, Multiprocessing y threads**
    - Ninguna de las arquitecturas de programación paralela OpenMP, MPI, Multiprocessing o threads se puede aplicar directamente a **SISD**, ya que todas ellas están diseñadas para sistemas de procesamiento paralelo en los que existen múltiples flujos de instrucciones y/o datos. Es importante destacar que, aunque SISD no es una arquitectura de procesamiento paralelo, es posible utilizar técnicas de procesamiento paralelo a nivel de algoritmo en sistemas **SISD**. Por ejemplo, algunos algoritmos pueden diseñarse para realizar cálculos en paralelo en varios núcleos de procesamiento en un sistema **SISD** mediante threads. Sin embargo, esto no cambia la clasificación del sistema como **SISD**, ya que solo hay un flujo de instrucciones y datos en cada núcleo de procesamiento.
    - Se puede implementar **SIMD** utilizando OpenMP, MPI, Multiprocessing y threads, aunque su implementación depende del contexto y del tipo de aplicación. **SIMD** se utiliza comúnmente en aplicaciones que procesan grandes conjuntos de datos, como señales y gráficos. OpenMP y MPI proporcionan herramientas para realizar operaciones **SIMD** en paralelo. En el caso de OpenMP, se pueden utilizar directivas específicas de **SIMD** para paralelizar un ciclo y ejecutar operaciones **SIMD** en el interior del bucle. Por otro lado, MPI ofrece rutinas y bibliotecas para realizar operaciones de procesamiento de datos en paralelo mediante el uso de **SIMD**.
    - La arquitectura **MISD** es rara en la actualidad, debido a que su uso no ofrece una mejora significativa en el rendimiento en la mayoría de las aplicaciones. De hecho, no se conocen implementaciones de **MISD** en sistemas de computación disponibles comercialmente. Por lo tanto, ninguna de las tecnologías de programación paralela como OpenMP, MPI, Multiprocessing o threads se aplica directamente a **MISD**.
    - Se puede utilizar varias tecnologías de programación paralela, como OpenMP, MPI, Multiprocessing o threads, para implementar la arquitectura MIMD, según el contexto y el tipo de aplicación. La arquitectura MIMD es comúnmente utilizada en aplicaciones que requieren procesamiento paralelo independiente. OpenMP y MPI son tecnologías que permiten la ejecución independiente de múltiples subprocesos y la comunicación entre procesos en diferentes nodos. Multiprocessing se refiere a la capacidad de procesar varias tareas en paralelo utilizando varios núcleos o procesadores. Los threads permiten la creación de varios hilos de ejecución dentro de un proceso, lo que permite la ejecución paralela de diferentes partes del código en diferentes núcleos o procesadores.

3. Con MPI y Multiprocessing despliegue verdad o falso si una palabra es palíndromo.
   Con MPI
   https://github.com/dazning/Ex_1_317/blob/main/palindromompi.c
   con Multiprocessing
   https://github.com/dazning/Ex_1_317/blob/main/palindromoMulti.ipynb

4. Con Multiprocessing y MPI genere la siguiente serie 0,1,1,2,3,5,8,13,…. Al menos unos 1000 elementos de la serie
   Con Multiprocessing
   https://github.com/dazning/Ex_1_317/blob/main/fibonacciMultiprocessing.ipynb
   Con MPI
   https://github.com/dazning/Ex_1_317/blob/main/fibompi.c

5. Con MPI y OpenMP realice el cálculo de Pi, mediante sumas sucesivas (unos 1000 elementos).
   Con MPI
   https://github.com/dazning/Ex_1_317/blob/main/pimpi.c

   Con OpenMP
   https://github.com/dazning/Ex_1_317/blob/main/piopenmp.c

6. Con OpenMP y MPI multiplique una matriz de 100x100.
   En MPI
   https://github.com/dazning/Ex_1_317/blob/main/p4_mpi.c

   En OpenMP
   https://github.com/dazning/Ex_1_317/blob/main/p4_openmp.c

   A

1	2	3	4	5	…	100
101	102					
					999	10000

   B

1	2	3	4	5	…	100
101	102					

					999	10000

7. Configure una máquina virtual con Linux debían y configure otra máquina Windows, para que se consulte a ambas mediante ssh (manual de como se hace). https://github.com/dazning/Ex_1_317/blob/main/Manual%20de%20conexi%C3%B3n%20SSH%20Windows%2010%20a%20Linux%20Debian%20y%20viceversa.pdf