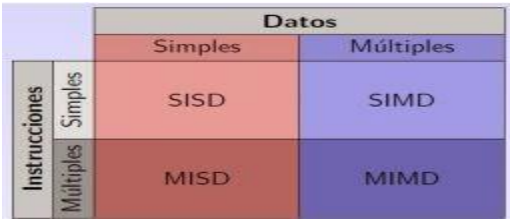
Nombre : NAVARRO QUISPE WILY RODRIGO
CI: 8420315
Materia: INF-317 Sistema en Tiempo Real y Distribuidos

Ejercicio 1

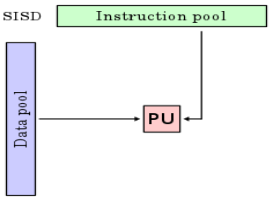
Explique cuál es la relación de la taxonomía de Flynn y cada una de las librerías utilizadas hasta el momento.

La taxonomía de Flynn es una clasificación para las computadoras con arquitectura paralela, propuesta por el profesor emérito de la Universidad de Stanford Michael J. Flynn, la cual clasifica a las mismas atendiendo a la cantidad de instrucciones y flujo de datos concurrentes en un instante de procesamiento.

La taxonomía de Flynn se clasifica de la siguiente manera.
- SISD:(Single Instruction Single Data)
- MISD:(Multiple Instruction Single Data)
- SIMD:(Single Instruction Multiple Data)
- MIMD:(Multiple Instruction Multiple Data)



Clasificación 1: SISD (Single Instruction Single Data)

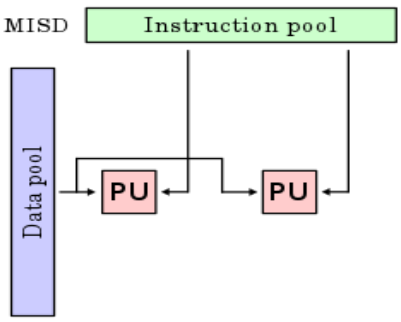


Esta clasificación se refiere a las computadoras tradicionales y secuenciales en las cuales una instrucción a la vez se ejecuta sobre un único dato cada ciclo de reloj. Los datos en cuestión se almacenan en una única memoria en la cual se usan técnicas como la segmentación para evitar errores de fragmentación interna. Un ejemplo sencillo de estas computadoras son los antiguos mainframe basados en la arquitectura de Von-Neumann.

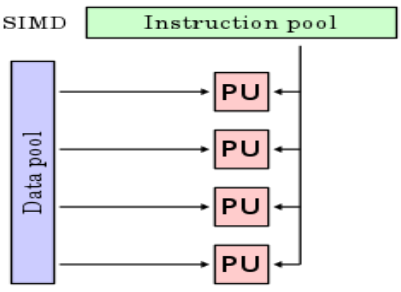
Características
- Son equipos con un solo procesador, que trabaja sobre un solo dato a la vez.
- A estos equipos se les llama también computadoras secuenciales.
- Flujo único de instrucciones.
- Flujo único de datos.

• Corresponde al modelo estructural básico, con un procesador de instrucciones y un
procesador de datos.
• Tiene una única vía de acceso a la memoria principal.

Clasificación 2: MISD:(Multiple Instruction Single Data)
Arquitectura que se refiere a múltiples instrucciones ejecutándose sobre un único dato. Comúnmente se considera esta arquitectura poco práctica ya que en tiempo de ejecución la efectividad del paralelismo requiere un múltiple flujo de datos y, además, el acceso concurrente a un mismo dato en memoria puede ocasionar que un CPU tenga que esperar a que el recurso(dato) esté disponible para poder acceder a él.

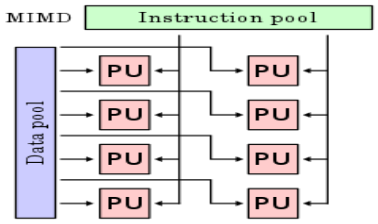


Clasificación 3: SIMD:(Single Instruction Multiple Data)
Esta arquitectura representa la ejecución de una misma instrucción sobre un conjunto de datos. La misma es comúnmente vista en ciclos de programación que ejecutan una misma instrucción una y otra vez sobre datos de un arreglo o conjunto de datos. En la arquitectura SIMD estos datos son procesados por múltiples CPU que ejecutan la
misma instrucción sobre una parte del conjunto o arreglo, cada uno, hasta llegar a procesar la totalidad de los mismos.
Algunas características son:
• SIMD (Single instruction multiple data) permite efectuar varias operaciones de cálculo con una sola instrucción.
• A los procesadores basados en esta arquitectura, se los conoce como procesadores
matriciales.
• Esta arquitectura nace debido a la necesidad de aplicar repetidamente una misma
operación en grupos datos diferentes como, Muestras contiguas de audio, matrices
de vídeo, etc.



Clasificación 4: MIMD:(Multiple Instruction Multiple Data)

Esta arquitectura representa a un conjunto de instrucciones que se ejecutan sobre
un conjunto múltiple de datos. La misma es muy usada hoy en día para explotar el paralelismo ya sea con memoria distribuida y memoria compartida o híbridos como los clústeres de computadoras.



CARACTERISTICAS

• Son sistemas con memoria compartida que permite ejecutar varios procesos simultáneamente (sistema multiprocesador)
• La diferencia con estos sistemas es que MIMD es asíncrono.
• No tiene un reloj central.
• Cuando las unidades de proceso reciben datos de una memoria no compartida estos
sistemas reciben el nombre de Múltiple SISD (MSISD).
• Los procesadores pueden ejecutar la misma o instrucción o diferentes instrucciones y
tener sus propios datos
• Diferentes elementos de información se asignan a diferentes procesadores
• Pueden tener memoria distribuida o compartida.

Librerias utilizadas hasta ahora:

omp.h (OpenMP): Esta biblioteca contiene instrucciones, rutinas y variables de entorno necesarias para trabajar con OpenMP (Open Multiprocesamiento), interfaz de programación para escritura. Programas paralelos en sistemas de memoria compartida. OpenMP Se utiliza para programar aplicaciones que utilizan paralelismo en hardware multinúcleo. Incluido omp.h C o C++, instrucciones disponibles Información sobre el compilador OpenMP como #pragma omp y funciones y variables que permiten monitorear y controlar la ejecución paralela.

import time: Esta librería proporciona funciones relacionadas con el tiempo. Algunas de las funciones más comunes de la librería time en Python son time.time(), que devuelve el tiempo actual en segundos desde el "epoch", time.sleep(segundos) que pausa la ejecución del programa durante la cantidad de segundos especificada, y time.strftime(formato) que permite formatear la representación de fechas y horas. Es útil para medir el tiempo de ejecución de un programa, introducir retrasos o realizar operaciones basadas en el tiempo.

from multiprocessing import Process: La librería multiprocessing en Python proporciona capacidades para la programación concurrente y paralela, permitiendo la ejecución de múltiples procesos de forma simultánea. Process es una clase en multiprocessing que se utiliza para crear y controlar procesos individuales. Con Process, puedes instanciar nuevos procesos, iniciarlos, esperar a que terminen, obtener información sobre su estado, entre otras operaciones.

rom multiprocessing import Pool: La clase Pool en el módulo multiprocessing de Python proporciona una manera conveniente de paralelizar la ejecución de funciones en un conjunto de datos mediante la asignación de los datos a varios procesos en un grupo (o "pool") de trabajadores. Por ejemplo, Pool.map() se usa para aplicar una función a cada elemento de una lista en paralelo.

import multiprocessing as mp: Esta línea importa el módulo multiprocessing y lo renombra como mp. Esto permite usar todas las funcionalidades de multiprocessing bajo el alias mp. Entonces, en lugar de escribir multiprocessing.funcion() se puede usar mp.funcion() para acceder a las funciones y clases de multiprocessing.

RELACIÓN DE LA TAXONOMÍA DE FLYNN Y CADA UNA DE LAS LIBRERÍAS UTILIZADAS HASTA EL MOMENTO.

SISD (Single Instruction, Single Data): Describe sistemas que ejecutan una sola instrucción sobre un solo dato en un momento dado. Esto se relaciona con tareas secuenciales y un único flujo de instrucción, como lo que se puede lograr con la librería time. Por ejemplo, la función time.sleep(segundos) detiene la ejecución de un programa durante un período de tiempo específico, lo que es un proceso secuencial.

SIMD (Single Instruction, Multiple Data): Hace referencia a sistemas en los que una sola instrucción se aplica a múltiples conjuntos de datos simultáneamente. multiprocessing en Python, particularmente el uso de multiprocessing.Pool o multiprocessing.Process, tiene cierta relación con esta categoría. Puede ejecutar funciones idénticas en paralelo en diferentes conjuntos de datos, similar al paralelismo SIMD.

MISD (Multiple Instruction, Single Data): Es un modelo teórico y poco común en aplicaciones prácticas. No hay muchas aplicaciones prácticas conocidas que se relacionen directamente con esta categoría.

MIMD (Multiple Instruction, Multiple Data): Se refiere a sistemas con múltiples procesadores ejecutando diferentes instrucciones en diferentes conjuntos de datos. En este caso, la librería omp.h (OpenMP) está más relacionada con el paradigma MIMD, ya que se utiliza para la programación en sistemas paralelos y distribuidos, permitiendo que múltiples hilos ejecuten diferentes secciones de código en paralelo, lo que encaja con el modelo MIMD.