**Pregunta 1**

**Estudiante:** Churqui Crispin Ivan Ramiro

**CI:** 6872496 LP

**1     Describa de manera teórica los siguientes conceptos: SISD, SIMD, MISD y MIMD. Indique además que lenguajes aplican a estos.**

**1. SISD (Single Instruction Stream, Single Data Stream)**

**Descripción**: En una arquitectura SISD, hay un solo flujo de instrucciones y un solo flujo de datos. Esta es la arquitectura tradicional de las computadoras secuenciales, donde una CPU ejecuta una instrucción a la vez en un solo conjunto de datos.

**Ejemplo de Lenguajes**: Lenguajes de programación como C, C++ y Java, cuando se utilizan en su forma más básica (sin paralelismo explícito).

**Ejemplo Práctico:** El siguiente ejemplo ilustra cómo funciona un programa en un entorno SISD, donde una única instrucción se aplica de manera secuencial a un conjunto de datos.

|  |
| --- |
| #include <stdio.h>  #define SIZE 5  int main() {  int array[SIZE] = {1, 2, 3, 4, 5};  int sum = 0;  // Sumar los elementos del arreglo  for (int i = 0; i < SIZE; i++) {  sum += array[i]; // Instrucción única aplicada a un solo conjunto de datos  }  printf("La suma de los elementos del arreglo es: %d\n", sum);  return 0;  } |

**2. SIMD (Single Instruction Stream, Multiple Data Streams)**

**Descripción**: En una arquitectura SIMD, una única instrucción se aplica a múltiples datos simultáneamente. Esto es útil en aplicaciones que requieren operaciones similares en grandes conjuntos de datos, como procesamiento de imágenes o simulaciones científicas.

**Ejemplo de Lenguajes**: Lenguajes que soportan extensiones específicas para SIMD, como C/C++ con intrínsecos SIMD (por ejemplo, SSE, AVX en x86) o mediante bibliotecas como OpenCL y CUDA (para procesamiento en GPUs).

**Ejemplo Práctico:** El siguiente ejemplo muestra cómo utilizar SIMD para realizar operaciones en paralelo sobre múltiples datos, aprovechando las capacidades de las instrucciones SSE en C.

|  |
| --- |
| #include <stdio.h>  #include <emmintrin.h> // Header para SSE2  #define SIZE 8  void sum\_arrays(const int \*a, const int \*b, int \*result) {  for (int i = 0; i < SIZE; i += 4) {  // Cargar 4 enteros de cada arreglo  \_\_m128i vec\_a = \_mm\_loadu\_si128((\_\_m128i\*)&a[i]);  \_\_m128i vec\_b = \_mm\_loadu\_si128((\_\_m128i\*)&b[i]);    // Sumar los vectores  \_\_m128i vec\_result = \_mm\_add\_epi32(vec\_a, vec\_b);    // Almacenar el resultado  \_mm\_storeu\_si128((\_\_m128i\*)&result[i], vec\_result);  }  }  int main() {  int a[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8};  int b[SIZE] = {9, 10, 11, 12, 13, 14, 15, 16};  int result[SIZE];  sum\_arrays(a, b, result);  printf("Resultado de la suma de arreglos:\n");  for (int i = 0; i < SIZE; i++) {  printf("%d ", result[i]);  }  printf("\n");  return 0;  } |

**3. MISD (Multiple Instruction Streams, Single Data Stream)**

**Descripción**: En una arquitectura MISD, múltiples flujos de instrucciones operan sobre un único flujo de datos. Esta arquitectura es menos común y se utiliza en aplicaciones muy específicas, como ciertos sistemas de redundancia y procesamiento de señales.

**Ejemplo de Lenguajes**: No hay un lenguaje específico que se asocie con MISD, ya que es más un concepto teórico. Sin embargo, sistemas diseñados para procesamiento redundante podrían implementarse utilizando C o ensamblador.

**Ejemplo Práctico:** El siguiente ejemplo ilustra el concepto de MISD al procesar un único dato de entrada a través de múltiples flujos de instrucciones (filtros) de manera secuencial. Aunque en la práctica, las arquitecturas MISD son raras, este código te da una idea de cómo se puede conceptualizar.

|  |
| --- |
| #include <stdio.h>  // Funciones de procesamiento  int filter1(int x) {  return x + 1; // Primer filtro: suma 1  }  int filter2(int x) {  return x \* 2; // Segundo filtro: multiplica por 2  }  int filter3(int x) {  return x - 3; // Tercer filtro: resta 3  }  int main() {  int input = 5; // Dato de entrada  int result1, result2, result3;  // Procesar el mismo dato de entrada a través de diferentes filtros  result1 = filter1(input); // Flujo de instrucciones 1  result2 = filter2(input); // Flujo de instrucciones 2  result3 = filter3(input); // Flujo de instrucciones 3  // Imprimir los resultados  printf("Resultado del filtro 1: %d\n", result1);  printf("Resultado del filtro 2: %d\n", result2);  printf("Resultado del filtro 3: %d\n", result3);  return 0;  } |

**4. MIMD (Multiple Instruction Streams, Multiple Data Streams)**

**Descripción**: En una arquitectura MIMD, hay múltiples flujos de instrucciones y múltiples flujos de datos. Esto permite que diferentes procesadores realicen diferentes tareas simultáneamente, cada uno con su propio conjunto de instrucciones y datos. Es la base de muchas arquitecturas de sistemas multiprocesador y clústeres.

**Ejemplo de Lenguajes**: Lenguajes que soportan programación paralela y concurrencia, como C (con pthreads), C++ (con bibliotecas de hilos y concurrencia), Java (con su API de concurrencia), y lenguajes de alto nivel como Python (usando multiprocessing o threading).

**Ejemplo Práctico:** El siguiente ejemplo muestra cómo utilizar múltiples hilos para procesar diferentes partes de un conjunto de datos al mismo tiempo, lo que es característico de la arquitectura MIMD. Cada hilo ejecuta un flujo de instrucciones diferente (sumar diferentes segmentos del arreglo) sobre múltiples datos.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <pthread.h>  #define SIZE 10  #define NUM\_THREADS 2  typedef struct {  int \*array;  int start;  int end;  int result;  } ThreadData;  void \*sum\_array(void \*arg) {  ThreadData \*data = (ThreadData \*)arg;  data->result = 0;  for (int i = data->start; i < data->end; i++) {  data->result += data->array[i];  }  pthread\_exit(NULL);  }  int main() {  int array[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  pthread\_t threads[NUM\_THREADS];  ThreadData thread\_data[NUM\_THREADS];  // Dividir el trabajo entre hilos  for (int i = 0; i < NUM\_THREADS; i++) {  thread\_data[i].array = array;  thread\_data[i].start = i \* (SIZE / NUM\_THREADS);  thread\_data[i].end = (i + 1) \* (SIZE / NUM\_THREADS);  pthread\_create(&threads[i], NULL, sum\_array, (void \*)&thread\_data[i]);  }  // Esperar a que terminen los hilos  int total\_sum = 0;  for (int i = 0; i < NUM\_THREADS; i++) {  pthread\_join(threads[i], NULL);  total\_sum += thread\_data[i].result;  }  printf("La suma total del arreglo es: %d\n", total\_sum);  return 0;  } |

**Resumen**

|  |  |  |
| --- | --- | --- |
| **Arquitectura** | **Descripción** | **Ejemplos de Lenguajes** |
| SISD | Un solo flujo de instrucciones y datos. | C, C++, Java |
| SIMD | Un flujo de instrucciones sobre múltiples datos. | C/C++ (con intrínsecos), OpenCL, CUDA |
| MISD | Múltiples flujos de instrucciones sobre un solo dato. | Teórico, implementaciones específicas |
| MIMD | Múltiples flujos de instrucciones y datos. | C, C++, Java, Python |

Estos conceptos son fundamentales para comprender cómo se organiza y optimiza el procesamiento en sistemas computacionales paralelos.