**Fundamento Teórico**

**Interconexión con Buses**

Los buses son canales de comunicación esenciales en la arquitectura de computadoras, ya que estos permiten la transferencia de datos, direcciones y señales de control entre los diferentes componentes. Según Phan y Nguyen (2023), un bus conecta todos los nodos del sistema a través de un protocolo que gestiona las solicitudes y concesiones de información, siendo esta una de las formas de interconexión más sencillos y ampliamente utilizados, tanto en sistemas de un solo núcleo como en los multinúcleos [1].

Por otra parte, Rogowski, los describe como una “autopista electrónica” que ofrece una vía compartida para la transmisión de datos, lo que los hace esenciales para el diseño lógico y la organización física de los sistemas computacionales [2].

**Elementos del diseño del bus**

El diseño de buses en sistemas modernos se caracteriza por elementos clave que definen su rendimiento y eficiencia. El ancho del bus determina la cantidad de datos que se pueden transferir de manera simultánea, aumentando el ancho de banda a costa de más líneas físicas en el chip. La velocidad del bus, medida en MHz o GHz, define cuántas transferencias pueden realizarse por segundo. Los protocolos de comunicación como AMBA AXI, establecen reglas para la gestión de acceso compartido, sincronización y modos de transferencia [3].

Según Trevor (2024), estos principios se aplican especialmente en SoCs modernos, donde estos elementos se adaptan a arquitecturas multicore y reconfigurables, donde buses avanzados y redes-on-chip (NoC) permiten conectar CPUs, aceleradores y memorias de forma escalable y eficiente [3].

**Tipos de buses**

**Bus de datos**

El bus de datos es un canal de comunicación que transfiere información entre la CPU, la memoria y los periféricos, manejando comúnmente 32 o 64 bits por ciclo, lo cual determina su ancho de banda. En los inicios de la PC, existían buses de datos en paralelo que conectaban directamente la memoria y los dispositivos, lo que obligaba a todos a operar a la misma velocidad y limitaba el rendimiento. Para dar solución a ese problema, se añadió un controlador que separa la CPU y la memoria de los periféricos, lo que mejoró la velocidad y comunicación entre los componentes. Hoy en día existen dos tipos principales de buses de datos: seriales y paralelos, como ejemplo de los seriales tenemos el USB, y ATA y SCSI para los paralelos [4].

**Bus de direcciones**

Wang (2021), describe el bus de direcciones como un canal unidireccional que la CPU y otros dispositivos utilizan para indicar una ubicación específica en la memoria o en algún dispositivo. El ancho del bus de direcciones determina la cantidad máxima que se puede direccionar, por ejemplo, un bus de 32 bits puede direccionar hasta 4 Gb de memoria. Algunos sistemas optimizan esto enviando la dirección en dos partes a través de la mitad de las líneas [4].

**Bus de control**

El bus de control es un conjunto de señales que viajan en ambas direcciones, las cuales la CPU utiliza para enviar comandos y recibir respuestas de otros componentes. También maneja interrupciones y se encarga de coordinar y sincronizar todas las operaciones entre la CPU, la memoria y los periféricos, asegurando que la información se transfiera en el momento y la forma correctos [4].

**Rol de los buses en la transferencia de datos entre componentes**

Los buses actúan como canales dedicados que interconectan los componentes de un sistema computacional, coordinando y optimizando el flujo de datos, con el objetivo de garantizar una comunicación eficiente. En arquitecturas especializadas como aceleradores DNN, estos soportan patrones críticos: uno a muchos para distribución de pesos, y de muchos a uno para recolección de resultados. Su diseño determina el rendimiento global, equilibrando ancho de banda, sincronización y escalabilidad, sobre todo en sistemas paralelos donde la transferencia de datos es esencial para el procesamiento acelerado [5].

**Procedimientos**

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

**Simulación en C# de comportamiento de una memoria caché**

**Código**

Texto

El contenido generado por IA puede ser incorrecto.

**Ejecución**

Texto

El contenido generado por IA puede ser incorrecto.

**Explicación**

Este programa simula cómo una CPU accede a datos desde la memoria RAM usando una caché pequeña.

* Se crea una caché de 4 bloques (como una "cajita" rápida donde la CPU guarda datos usados recientemente).
* La CPU intenta leer 10 direcciones de memoria (como 0, 4, 8, 12, 0, 4, 16, 20, 0, 4).
* En cada acceso:
* Si el dato está en caché (HIT): La CPU lo lee al instante.
* Si no está (MISS): La CPU lo trae de la RAM (más lento) y lo guarda en caché para después.

1. **Inicialización:**

La caché empieza vacía (todos los bloques valen -1).

1. **Acceso a memoria:**

La CPU quiere leer la dirección 0:

* MISS (la caché está vacía).
* Guarda el dato en la caché.

Luego lee 4:

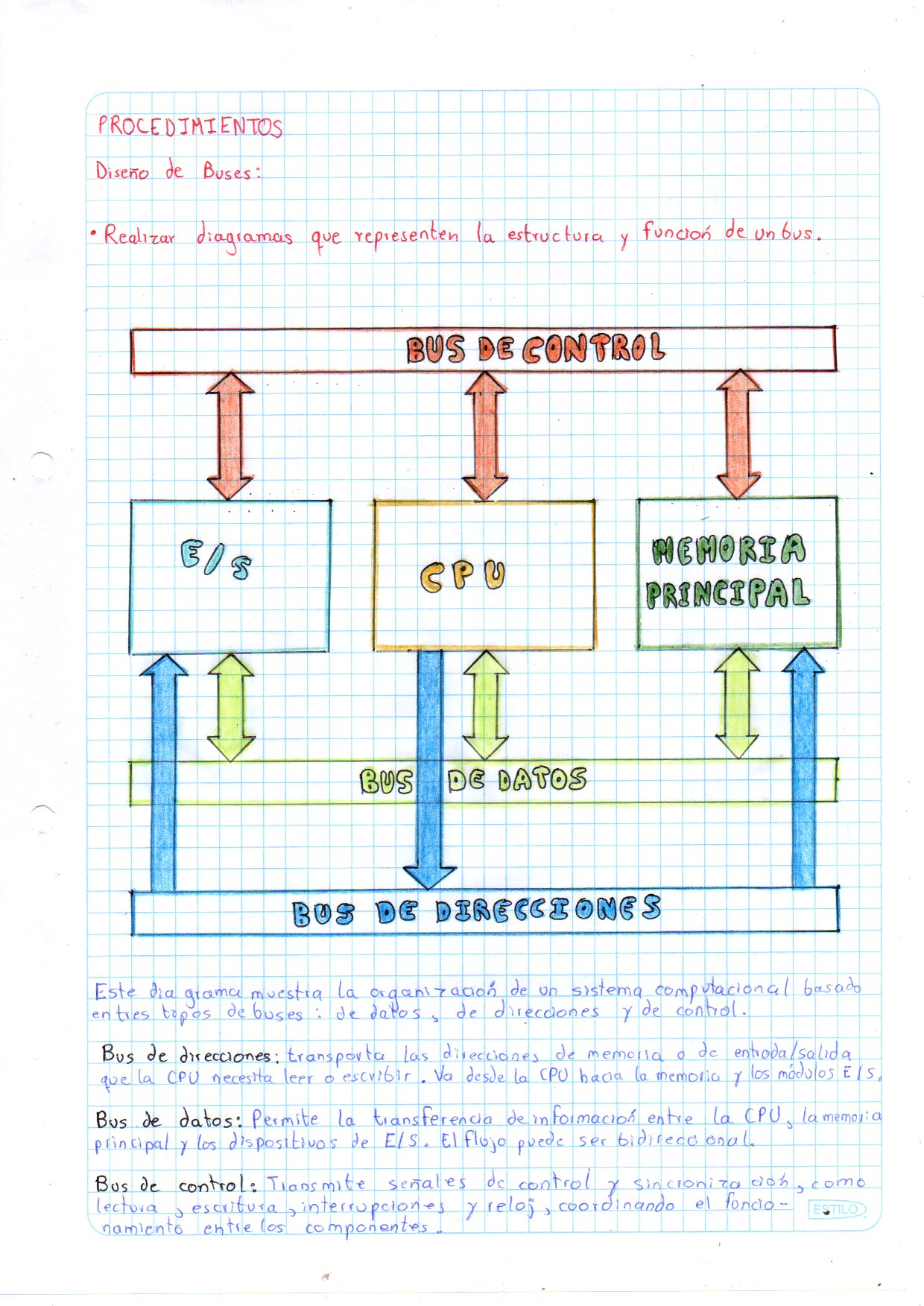
* MISS (nuevo dato).
* Lo guarda.

Cuando repite 0:

* HIT (ya estaba en caché).

1. **Resultados:**

Muestra cuántos accesos fueron rápidos (HIT) y cuántos lentos (MISS).



Tabla

El contenido generado por IA puede ser incorrecto.

**Bibliografía**

[1] C. V. Phan and T. D. Nguyen, *Context-Aware Systems and Applications*, vol. 475. Cham: Springer Nature Switzerland, 2023. doi: 10.1007/978-3-031-28816-6.

[2] S. J. Rogowski, “Bus,” in *Encyclopedia of Computer Science*, GBR: John Wiley and Sons Ltd., 2003, pp. 165–167. doi: 10.5555/1074100.1074184.

[3] T. E. Carlson, “Bus and Memory Architectures,” in *Handbook of Computer Architecture*, Singapore: Springer Nature Singapore, 2024, pp. 201–212. doi: 10.1007/978-981-97-9314-3\_68.

[4] S. P. Wang, *Computer Architecture and Organization*, 1st ed. Singapore: Springer Singapore, 2021. doi: 10.1007/978-981-16-5662-0.

[5] B. Tiwari, M. Yang, X. Wang, and Y. Jiang, “Data Streaming and Traffic Gathering in Mesh-based NoC for Deep Neural Network Acceleration,” *CoRR*, vol. abs/2108.02569, 2021, doi: 10.48550/arXiv.2108.02569.