FACULTAD DE CIENCIAS DE LA COMPUTACIÓN Y DISEÑO DIGITAL

INTEGRANTES

HERRERA MORAN GILMAR JAVIER

MENDOZA PARRAGA ANDY JOHEL

ZAMORA AGUILAR RONALDO WILFRIDO

CURSO

2DO SOFTWARE “B”

GRUPO

F

MATERIA

ARQUITECTURA DE COMPUTADORAS

TEMA

ARITMÉTICA DE LA COMPUTADORA Y ÁLGEBRA DE BOOLE

ÍNDICE

OBJETIVO 4

FUNDAMENTO TEÓRICO 4

1. Aritmética Binaria y Decimal 4

1.1. Aritmética Binaria 4

1.1.1. Conversión entre sistemas binario y decimal 4

1.1.2. Importancia de la base 2 en el procesamiento de datos en computadoras 5

1.1.3. Operaciones aritméticas básicas en binario 5

1.1.3.1. Suma en binario 5

1.1.3.2. Resta en binario 5

1.1.3.3. Multiplicación en binario 6

1.1.3.4. División en binario 6

1.2. Aritmética decimal 7

1.2.1. ¿Para qué sirve la Aritmética Decimal? 8

2. Álgebra de Circuitos Digitales y Función Boole 8

2.1. Función Boole 8

2.1.1. Funciones y Operadores Booleanos. 9

2.2. Álgebra de circuitos digitales 9

2.2.1. Análisis de álgebra booleana aplicada a circuitos lógicos y digitales 9

3. Circuitos Lógicos y Sistemas Digitales 10

3.1. Circuitos secuenciales 10

3.2. Circuitos combinacionales 10

3.3. Sistemas combinacionales básicos 11

3.3.1. Sumadores 11

3.3.2. Restadores 12

3.3.3. Multiplexores 13

4. Microprocesador y Microcontroladores 14
4.1. Microprocesadores 14
4.1.1. Velocidad de procesamiento y ciclo de instrucciones 14
4.2. Microcontroladores 15
4.2.1. Velocidad de procesamiento y ciclo de instrucciones 15
PROCEDIMIENTOS 15
A. Operaciones de Aritmética Binaria y Decimal 15
B. Ejercicios de Álgebra Booleana 20
C. Diseño de Circuitos Combinacionales 25
D. Estudio de Microprocesadores y Microcontroladores 26
CONCLUSIÓN 29
BIBLIOGRAFÍA 29
ANEXO 33

ARITMÉTICA DE LA COMPUTADORA Y ÁLGEBRA DE BOOLE

OBJETIVO

Aplicar conceptos de aritmética binaria y decimal, álgebra booleana y circuitos lógicos para comprender el funcionamiento de los microprocesadores, microcontroladores y sistemas digitales.

FUNDAMENTO TEÓRICO

1. Aritmética Binaria y Decimal

1.1. Aritmética Binaria

La base 2, también conocida como sistema binario, es un sistema binario, es un sistema numérico que utiliza únicamente dos dígitos: 0 y 1. Este sistema es fundamental en el ámbito de la computación, ya que las computadoras operan internamente utilizando circuitos electrónicos que pueden estar en uno de dos estados: encendido (1) o apagado (0). Esta representación binaria permite a las computadoras procesar y almacenar datos de manera eficiente [1].

1.1.1. Conversión entre sistemas binario y decimal

La conversión entre decimal (base 10) y binario (base 2) es crucial para comprender cómo las computadoras procesan los datos. Para convertir de decimal a binario, divida el número entre 2 repetidamente, anotando los residuos, hasta llegar a 0. Luego, lea los residuos de abajo a arriba [2], [3].

Por ejemplo, 13 en decimal se convierte a binario de la siguiente manera:

$13 \div 2 = 6$, residuo 1

$6 \div 2 = 3$, residuo 0

$3 \div 2 = 1$, residuo 1

$1 \div 2 = 0$, residuo 1

Resultado: **1101**.

Para convertir de binario a decimal, se multiplica cada bit por 2 elevado a su posición, de derecha a izquierda. Ejemplo:

$$1101 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = 8 + 4 + 0 + 1 = \mathbf{13}.$$

Estos procesos son esenciales para traducir la información entre humanos y máquinas.

1.1.2. Importancia de la base 2 en el procesamiento de datos en computadoras

La importancia del sistema binario radica en su simplicidad y en la facilidad con la que se puede implementar en hardware. Cada bit (dígito binario) puede ser representado por un estado eléctrico, lo que permite realizar operaciones lógicas y aritméticas a gran velocidad. Además, el uso del sistema binario facilita la codificación de diferentes tipos de datos, como números, caracteres y multimedia, en formatos que las computadoras pueden entender y manipular [2], [4].

1.1.3. Operaciones aritméticas básicas en binario

Las operaciones aritméticas en binario son similares a las que se realizan en el sistema decimal, pero se llevan a cabo utilizando solo los dígitos 0 y 1. En los siguientes puntos se describen las operaciones básicas como: suma, resta, multiplicación y división, incluyendo el manejo de Carry (llevar) y Borrow (préstamo) [3].

1.1.3.1. Suma en binario

Es una operación fundamental en la aritmética de los sistemas numéricos binarios, que son la base de la computación moderna. En el sistema binario, solo se utilizan dos dígitos: 0 y 1. Esto contrasta con el sistema decimal, que utiliza diez dígitos (0-9). La suma de binario sigue principios similares a la suma en decimal, pero con algunas diferencias clave debido a la naturaleza del sistema. La suma en binario es esencial en la computación, ya que todos los datos y operaciones en las computadoras se representan en binario. Las operaciones aritméticas, como la suma se implementan en hardware a través de circuitos lógicos, como sumadores, que realizan estas operaciones de manera eficiente [2].

Reglas básicas de suma binaria

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 10$$

(es decir, 0 con acarreo 1)

1.1.3.2. Resta en binario

Es una operación aritmética fundamental que se utiliza en el sistema numérico binario, el cual es la base de la computación moderna. Al igual que en el sistema decimal, la resta en binario implica la sustracción de un número de otro. Sin embargo, debido a la naturaleza del sistema binario, la resta presenta algunas particularidades que son importantes de entender [1].

Esta operación desempeña un papel importante dentro del funcionamiento de los sistemas digitales, ya que las computadoras procesar la información utilizando este formato. Para ello, se emplean componentes electrónicos especializados, como los restadores, que permiten llevar a cabo esta operación con rapidez y precisión. Esta operación es especialmente relevante en tareas como el control automático, el tratamiento de señales y el manejo de información [1], [3].

Reglas básicas de resta

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$0 \quad - 1 = 1 \text{ (con Borrow de la siguiente posición)}$$

1.1.3.3. Multiplicación en binario

Es una operación aritmética fundamental que se utiliza en el sistema numérico binario, que es la base de la computación. Al igual que en el sistema decimal, la multiplicación en binario implica la combinación de dos números (multiplicando y multiplicador) para obtener un producto. Sin embargo, debido a la naturaleza del sistema binario, la multiplicación presenta algunas particularidades que son importantes de entender [5].

La operación de multiplicar en binario es esencial en el ámbito de la computación, ya que las computadoras ejecutan todas sus operaciones aritméticas utilizando este sistema numérico. Los circuitos lógicos, como los multiplicadores, permiten realizar estas operaciones de forma eficiente. Además, la multiplicación juega un papel clave en algoritmos de control, procesamiento de señales y manejo de datos [3], [5].

Reglas básicas de multiplicación

$$0 * 0 = 0$$

$$0 * 1 = 0$$

$$1 * 0 = 0$$

$$1 * 1 = 1$$

1.1.3.4. División en binario

Es una operación aritmética fundamental que se utiliza en el sistema numérico binario, que es la base de la computación moderna. Al igual que en el sistema decimal, la división en binario implica la separación de un número en partes iguales según otro número. Sin embargo, debido

a la naturaleza del sistema binaria, la división presenta algunas particularidades que son importantes de entender [2], [5].

La división binaria es una operación esencial dentro de los sistemas computacionales, ya que las máquinas trabajan internamente con números binarios. Para ejecutar esta función de manera óptima, se utilizan dispositivos lógicos llamados divisores. Esta operación resulta especialmente útil en aplicaciones como el procesamiento de datos, los sistemas de control y el análisis de señales digitales [4].

Reglas de división

$$0 / 1 = 0$$

$$1 / 1 = 1$$

$$0 / 0 = \text{ES INDEFINIDO}$$

$$1 / 0 = \text{ES INDEFINIDO}$$

1.2. Aritmética decimal

La aritmética decimal es un conjunto de métodos, algoritmos y circuitos diseñados para procesar matemáticamente (sumas, restas, multiplicaciones, divisiones...) números que se presentan en base 10 con precisión decimal y sin errores de conversión a binario. Esta precisión es clave en aplicaciones de finanzas, contabilidad y comercio, donde error en el redondeo pueden dar lugar a graves problemas. Los ordenadores, sin embargo, trabajan internamente en binario (base 2), por lo que convertir los números decimales a binario con precisión suele llevar a errores de redondeo (y fallos de precisión). Para evitar estos problemas nace la aritmética decimal binaria (Binary Decimal Arithmetic) que parte de la idea de representar y operar números decimales directamente en un sistema binario con alta precisión [6].

Para ello propone el diseño de formatos de representación de números especiales como Decimal Coded Binary (BCD) o formatos de número real decimal, junto con métodos y hardware específicos que aseguren el resultado correcto y lo hagan de manera eficiente. Los dos libros, Hardware–Software Co-Design for Decimal Multiplication y Hardware Designs for Decimal Floating-Point Addition and Related Operations, se centran en cómo implementar aritmética decimal de forma eficiente ya sea a través de diseños híbridos hardware-software o bien mediante hardware dedicado. La idea es acelerar los cálculos, pero siempre asegurando que los resultados sean precisos y exactos, algo esencial en áreas donde tener precisión en la base es fundamental [6], [7].

1.2.1. ¿Para qué sirve la Aritmética Decimal?

La aritmética decimal es útil para hacer cálculos precisos con números en base 10, como los que solemos usar en la vida diaria, asegurándonos de que no se pierda su exactitud por errores de redondeo. Es esencial en situaciones en las que se requiere precisión: a la hora de hacer cálculos financieros y en contabilidad o comercio, en educación o en el diseño de hardware digital a comparación de los números de base 2 en binario [1], [7].

En computadoras y tecnología

Para un sistema digital, trabajar con el sistema numérico binario es fácil. Sin embargo, el convertir un número que pertenece al sistema de numeración decimal al binario puede producir errores. La Binary Decimal Arithmetic es una forma de que un número que pertenece al ámbito de los números decimales se represente correctamente en el ámbito de los números binarios y de que las operaciones de suma, resta, etc. se realicen sobre el número decimal real, sin errores. Esto es fundamental en banca, bases de datos de dinero y en cálculos científicos donde no puede haber un error decimal. El libro Hardware Designs for Decimal Floating-Point Addition and Related Operations habla de cómo es posible tener hardware especializado que realice operaciones como suma, resta, redondeo [7].

En la educación matemática

Desde la escuela, como se muestra en el estudio Mejora del rendimiento matemático en Primaria, la comprensión de fracciones y números decimales es crucial para mejorar el rendimiento en esta asignatura. No solo es imprescindible para las matemáticas el dominio de la aritmética decimal, sino que esta también favorece el razonamiento lógico, la capacidad de resolver problemas y la competencia matemática. La enseñanza personalizada de fracciones y decimales permite a los estudiantes posibilitar emplear los fundamentos sobre los que se aplican en ciencia, tecnología, ingeniería y matemáticas [8].

2. Álgebra de Circuitos Digitales y Función Boole

2.1. Función Boole

Una función booleana es una relación matemática que convierte variables booleanas en una salida booleana, estás salidas solo pueden tener dos estados falso o verdadero, lo cuales se representan como 0 y 1 respectivamente, y son utilizados para simular el comportamiento lógico de circuitos digitales [9].

Las funciones booleanas también se representan mediante expresiones algebraicas, ya sea suma de productos o producto de sumas, tablas de verdad, circuitos con compuertas lógicas o

diagramas. Cada una de estas maneras de representar a las funciones booleanas tienes sus ventajas dependiendo del contexto que sea requerido. Estas funciones también se pueden utilizar como operaciones lógicas suma, resta, multiplicación y comparación [10].

2.1.1. Funciones y Operadores Booleanos.

En el álgebra booleana encontramos los operadores booleanos, que son los operadores fundamentales para aplicar en variables binarias. Los más conocidos son:

- AND (∧): devuelve 1 si ambas entradas son 1.
- OR (∨): devuelve 1 si al menos una de las entradas son 1.
- NOT (¬): invierte el valor de la entrada.

De estos operadores se derivan otros operadores como XOR, NAND, NOR.

Las leyes del álgebra booleana, ya sea conmutativa, distributiva, De Morgan o nula, permiten simplificar expresiones booleanas sin cambiar su resultado booleano, verdadero o falso. Esto es indispensable para optimizar circuitos booleanos [11].

2.2. Álgebra de circuitos digitales

El álgebra de circuitos digitales es importante para el uso de expresiones booleanas y sirve para diseñar y optimizar circuitos lógicos. Gracias a esta se pueden mostrar requerimientos funcionales en diseños con puertas lógicas básicas (AND, OR, NOT) o compuertas compuestas (NAND, XOR, NOR) [12].

Un circuito digital puede ser combinacional, este depende de las entradas actuales, o secuencial, este no solo depende de las entradas actuales, sino que también de estados anteriores. Para los circuitos combinacionales, el álgebra booleana sirve para definir la lógica, lo cual nos permite construir circuitos más eficientes, mejorar el control del flujo de datos y optimizar la cantidad de puertas lógicas utilizadas [13].

2.2.1. Análisis de álgebra booleana aplicada a circuitos lógicos y digitales

Este análisis nos permite diseñar sistemas con una alta precisión. Gracias a las tablas de verdad podemos escribir una función booleana que describe el comportamiento requerido en el circuito, luego de esto se utilizan técnicas para la simplificación o reducción de la función booleana para poder adaptarla al hardware usando la menor cantidad de componentes posibles [14].

Los mapas de Karnaugh también son fundamentales para simplificar funciones booleanas, ya que nos ayuda a organizar los valores de la tabla de verdad de manera visual en una cuadricula,

donde cada una de sus celdas representa una combinación de sus entradas, luego se procede a agrupar la mayor cantidad de unos posibles con el fin de reducir la cantidad de operaciones lógicas necesarias en una función. Con lo cual de obtienes circuitos más pequeños, rápidos y eficientes. A diferencia del método algebraico, los mapas de Karnaugh nos ayudan a facilitar la simplificación porque al ser este un medio visual, la tarea se vuelve más intuitiva [15].

3. Circuitos Lógicos y Sistemas Digitales

Los circuitos digitales son sistemas electrónicos que operan con señales binarias, 0 y 1 o de voltaje bajo y voltaje alto. Se basan en el álgebra booleana, lo que permite ejecutar operaciones lógicas y aritméticas a través de compuertas lógicas [16]. Estos circuitos son fundamentales en sistemas digitales como computadoras, teléfonos móviles y dispositivos de comunicación, y han evolucionado desde circuitos integrados simples (CI) a chips de alta complejidad como los VLSI y ULSI [17].

3.1. Circuitos secuenciales

Los circuitos secuenciales son aquellos en los que su salida no solo depende de lo que entra en ese momento, sino también de lo que les pasó antes, es decir, el estado anterior del sistema. Estos guardan información internamente, lo que les da una memoria y hace que su comportamiento cambie con el tiempo. Esto se logra utilizando componentes como Flip-Flops, que retienen un valor hasta que una señal de reloj indique que debe cambiar [18].

Los Flip-Flops se clasifican según su comportamiento: El Flip-Flop SR actúa como una memoria de 1 bit con entradas SET y RESET. RESET pone la salida en 0, y SET la pone en 1. Los D Flip-Flops conservan el valor de su entrada mientras está sincronizado con una señal de reloj, y los JK Flip-Flops son similares a los SR, pero tienen la ventaja de incluir una entrada de reloj y compuertas adicionales que evitan resultados incorrectos. Puede operar en cuatro modos diferentes: establecer la salida 1, a 0, mantener su estado anterior o alternar (toggle) entre 0 y 1 [19].

3.2. Circuitos combinacionales

Los circuitos combinacionales se forman interconectando varias puertas lógicas. Para que estas funcionen sin problemas deben ser concatenables. La concatenabilidad significa que la señal de salida de una puerta debe ser compatible con la entrada de otras puertas o incluso de circuitos secuenciales, lo que permite que se realicen operaciones lógicas continuas. Para evitar conflictos, tanto las entradas como las salidas deben usar la misma magnitud física. Por

ejemplo, si se emplean niveles de voltaje para codificar las entradas, las salidas también deben expresarse mediante voltajes para asegurar una conexión directa entre los componentes [20].

3.3. Sistemas combinacionales básicos

3.3.1. Sumadores

Un sumador es un circuito combinacional constituido por puertas lógicas que recibe dos entradas binarias (o tres entadas, si se incluye un acarreo de entrada o carry-in), y su resultado es el bit de la suma, y un posible bit de acarreo [21].

Existen algunos tipos de sumadores. El sumador medio (Half Adder) realiza la suma de dos bits sin considerar el acarreo de entrada, mientras que el sumador completo (Full Adder) amplia esta función sumando tres bits, incluyendo el acarreo de entrada. Según Nagaraj et al. (2025), el Full Adder puede ser optimizado mediante técnicas como ECRAAL con el objetivo de reducir el consumo energético en diseño VLSI. A partir de este último, nacen estructuras más complejas como el Sumador con acarreo en cascada (Ripple Carry Adder), el Sumador con anticipación de acarreo (Carry Lookahead Adder) y Sumadores aproximados (Approximate Adder), cada uno con ventajas en velocidad, complejidad o eficiencia energética [21].

Ejemplo de sumadores:

Tabla 1. Tabla de Verdad Sumador

A	B	S	Cout
0	0	0	0
0	0	1	0
0	1	1	0
0	1	0	1

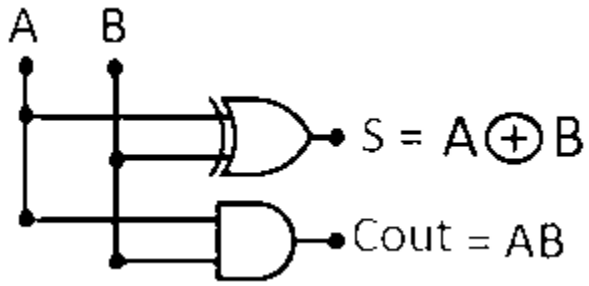


Figura 1. Sumador

3.3.2. Restadores

Los restadores son circuitos combinacionales que realizan la operación de resta binaria, ya sea entre dos bits individuales o entre números binarios de múltiples bits. Hay versiones simples como el restador medio (Half Subtractor), que resta dos bits sin préstamo previo, y el restador completo (Full Subtractor), que si considera un bit de préstamo de entrada (borrow in) [22].

Según Sanadhya y Sharma (2020), los restadores completos pueden implementarse eficientemente usando lógica adiabática, lo que permite disminuir tanto el consumo energético como el retardo en comparación con diseños CMOS comunes [23].

Ejemplo de restadores:

Tabla 2. Tabla de Verdad

A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

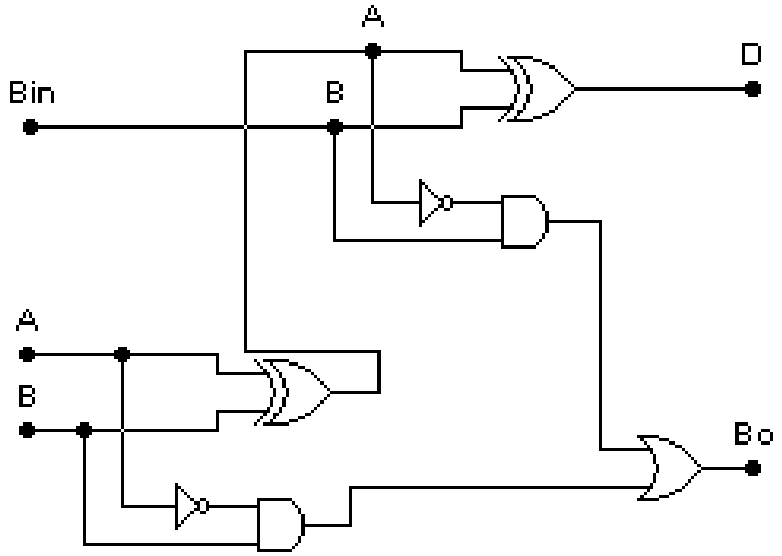


Figura 2. Restador

3.3.3. Multiplexores

Un multiplexor es un circuito lógico combinacional que permite la selección de una única salida entre múltiples señales de entrada, utilizando un conjunto de líneas de control binarias. Su función principal es optimizar el uso de recursos al permitir que varios datos compartan un mismo canal de transmisión o línea de procesamiento. Los tipos más comunes de multiplexores son los de 2:1, 4:1 y 8:1, donde el número que sigue a los dos puntos indica cuántas entradas de datos son supervisadas por las líneas de control [24].

Según Bansal, Singh y Sharma (2021), estas estructuras son esenciales en sistemas digitales como microprocesadores y redes de comunicación, debido a que permiten reducer el número de líneas físicas necesarias y optimizar el flujo de datos en aplicaciones embebidas [24].

Ejemplo de multiplexores:

Tabla 3. Tabla de Verdad Multiplexor

S1	S0	A	B	C	D	Y
0	0	x	X	X	0	0
0	0	X	X	X	1	1
0	1	X	X	0	X	0
0	1	X	X	1	X	1
1	0	X	0	X	X	0
1	0	X	1	X	X	1
1	1	0	X	X	X	0
1	1	1	x	X	X	1

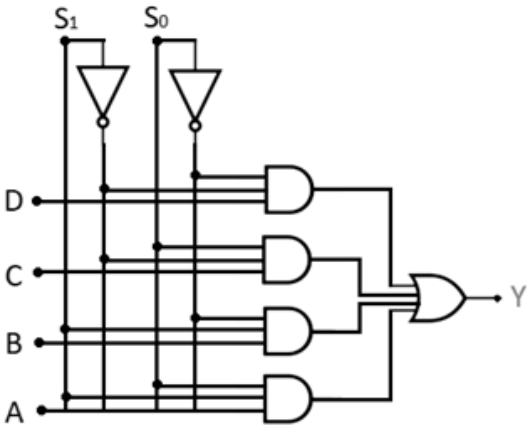


Figura 3. Multiplexor

4. Microprocesador y Microcontroladores

4.1. Microprocesadores

Un microprocesador es un circuito integrado de tamaño reducido, compuesto por miles o millones de transistores, el cual tiene como función principal recibir datos de entrada, procesarlos siguiendo un conjunto de instrucciones específicas y entregar resultados como salida. Se encuentra en el núcleo de la CPU de una computadora, así como en otros sistemas electrónicos, siendo el responsable de ejecutar operaciones aritméticas, lógicas y de control con gran velocidad y precisión [25].

Para Abdullayevich (2020), el microprocesador es el elemento central del sistema computacional, encargado de coordinar y ejecutar los procesos internos de la computadora [26].

Según Antonov y Krasnyuk (2021), el microprocesador está constituido por diversos componentes que operan en conjunto para ejecutar instrucciones de manera eficaz. Entre estos se destaca la unidad de control que interpreta y dirige la ejecución de instrucciones, la unidad aritmético-lógica (ALU por sus siglas en inglés) encargada de las operaciones matemáticas y lógicas, registros internos que almacenan datos temporalmente, y la memoria caché que acelera el acceso a información frecuente [27].

4.1.1. Velocidad de procesamiento y ciclo de instrucciones

La velocidad de procesamiento de un microprocesador depende de varios factores que dependen uno de otros, como la frecuencia de reloj, la eficiencia de instrucciones y la arquitectura interna [28].

Suarez, Almeida y Blanco (2024), determinaron que, aunque las implementaciones de la arquitectura RISC-V puede presentar un menor consumo promedio de energía en comparación con ARM, esto no quiere decir que sea superior en el rendimiento por vatio. Es decir, una menor potencia consumida no siempre se traduce como mayor eficiencia energética si el procesador no puede ejecutar instrucciones de manera eficiente [28].

El ciclo de instrucciones en los microprocesadores consta de cuatro etapas: búsqueda de la instrucción en memoria, decodificación para interpretar la operación, ejecución de la instrucción, acceso a memoria si es necesario y escritura del resultado en registros. Nikolic et al. (2022), señalan que la duración y eficiencia de este ciclo afectan directamente a la velocidad de procesamiento, debido a que cuanto más corto sea un ciclo, más instrucciones se pueden ejecutar por segundo, además que el uso de técnicas como la segmentación o pipelining,

mejoran el rendimiento al procesar múltiples instrucciones de forma simultánea en diferentes fases del ciclo [29].

4.2. Microcontroladores

Los microcontroladores son circuitos integrados que tienen en un solo chip la CPU, memoria y periféricos. A diferencia de los microprocesadores, los microcontroladores están diseñados para controlar tareas específicas, lo que lo los hace ideales para aplicaciones autónomas [30].

Un microcontrolador tiene 3 tipos de memorias: memoria de programa (Flash), memoria de datos (SRAM) y memoria de configuración (EEPROM). También tienes sus periféricos como convertidores analógico-digital (ADC), puertos GPIO, módulos de temporizador, contadores y mecanismos de comunicación serial. Gracias a estas características el microcontrolador puede interactuar eficientemente con el entorno físico y procesar datos en tiempo real [31].

La arquitectura de los microcontroladores también viene optimizada para reducir el consumo energético y con el mismo poderío funcional. Esto gracias a que tienen modos de bajo consumo, interrupciones configurables y escalado dinámico de frecuencias [32].

4.2.1. Velocidad de procesamiento y ciclo de instrucciones

La velocidad de procesamiento de los microcontroladores depende directamente de su frecuencia de reloj y de la eficiencia del ciclo de instrucciones. Cada una de las instrucciones tiene un ciclo básico que incluye la fase de búsqueda (Fetch), decodificación (decode) y ejecución (execute). Todo esto depende del número de ciclos de reloj que requiere y de la arquitectura interna del microcontrolador [33].

Otro aspecto clave es el uso de interrupciones, estas permiten que el microcontrolador responda a los eventos externos sin necesidad de consultar constantemente el estado de los dispositivos conectados. El análisis del ciclo de instrucción también es fundamental en las aplicaciones donde el tiempo real es un factor importante, ya que el cálculo preciso de cuantos ciclos consume cada una de las operaciones, con esto se puede prever el rendimiento del sistema y ajustar temporizadores de control en consecuencia [34].

PROCEDIMIENTOS

A. Operaciones de Aritmética Binaria y Decimal

Realizar ejercicios de suma, resta, multiplicación y división en binario y decimal.

Nombres: Herrera Novoa Gilmar Javier

Curso: 2do Software "B"

Fecha: 24/07/2023

- Realizar ejercicios de suma, resta, multiplicación y división en binario y decimal
- Comparar resultados obtenidos en ambos sistemas para evaluar la precisión

- Suma decimal y binario

$$\begin{array}{r} 25_{10} \\ +\ 17_{10} \\ \hline 42_{10} \end{array}$$

$25_{10} \rightarrow 11001_2$

$17_{10} \rightarrow 10001_2$

$42_{10} \rightarrow 101010_2$

$$\begin{array}{r} 11001_2 \\ +\ 10001_2 \\ \hline 101010_2 \end{array}$$

$$\begin{array}{r} {\scriptstyle 2} \\ 135_{10} \\ +\ 89_{10} \\ \hline 47_{10} \\ \hline 271_{10} \end{array}$$

$135_{10} \rightarrow 10000111_2$

$89_{10} \rightarrow 01011001_2$

$47_{10} \rightarrow 00101111_2$

$271_{10} \rightarrow 100001111_2$

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ 1\ 1\ 1\ 1} \\ 10000111_2 \\ +\ 01011001_2 \\ \hline 00101111_2 \\ \hline 100001111_2 \end{array}$$

- Resta decimal y binario

$$\begin{array}{r} 50_{10} \\ -\ 23_{10} \\ \hline 27_{10} \end{array}$$

$50_{10} \rightarrow 110010_2$

$23_{10} \rightarrow 010111_2$

$27_{10} \rightarrow 011011_2$

$$\begin{array}{r} 110010_2 \\ -\ 010111_2 \\ \hline 011011_2 \end{array}$$

$$\begin{array}{r} 200_{10} \\ -\ 75_{10} \\ \hline 125_{10} \end{array}$$

$200_{10} \rightarrow 11001000_2$

$75_{10} \rightarrow 01001011_2$

$125_{10} \rightarrow 01111101_2$

$$\begin{array}{r} 11001000_2 \\ -\ 01001011_2 \\ \hline 01111101_2 \end{array}$$

Multiplicación decimal y binario

$$
\begin{array}{r}
13_{10} \\
\times\ 6_{10} \\
\hline
78_{10}
\end{array}
$$

$13_{10} \rightarrow 1101_2$
$6_{10} \rightarrow 0110_2$
$78_{10} \rightarrow 1001110_2$

$$
\begin{array}{r}
1101_2 \\
\times\ 0110_2 \\
\hline
0000 \\
1101 \\
1101 \\
0000 \\
\hline
1001110_2
\end{array}
$$

$$
\begin{array}{r}
23 \\
\times\ 19 \\
\hline
207 \\
+\ 230 \\
\hline
437
\end{array}
$$

$23_{10} \rightarrow 10111_2$
$19_{10} \rightarrow 10011_2$
$437_{10} \rightarrow 110110101_2$

$$
\begin{array}{r}
10111_2 \\
\times\ 10011_2 \\
\hline
10111 \\
10111 \\
+\ 00000 \\
00000 \\
10111 \\
\hline
110110101_2
\end{array}
$$

División decimal y binario

$$
\begin{array}{r|l}
156 & 12 \\
\hline
12 & 13 \\
\hline
036 & \\
36 & \\
\hline
00 &
\end{array}
$$

$156_{10} \rightarrow 10011100_2$
$12_{10} \rightarrow 00001100_2$
$13_{10} \rightarrow 1101_2$

$$
\begin{array}{r|l}
10011100 & 00001100 \\
\hline
1100 & 1101 \\
\hline
001110 & \\
-\ 1100 & \\
\hline
001100 & \\
1100 & \\
\hline
(0000) &
\end{array}
$$

$$
\begin{array}{r|l}
45 & 5 \\
\hline
45 & 9 \\
\hline
00 &
\end{array}
$$

$45_{10} \rightarrow 101101_2$
$5_{10} \rightarrow 00101_2$
$9_{10} \rightarrow 1001_2$

$$
\begin{array}{r|l}
101101 & 00101 \\
\hline
-\ 101 & 1001 \\
\hline
000101 & \\
101 & \\
\hline
000000 &
\end{array}
$$

Suma binario y decimal

$$\begin{array}{r} 1010_2 \\ +\ 1101_2 \\ \hline 10111_2 \end{array}$$

$1010_2 \rightarrow 10_{10}$

$1101_2 \rightarrow 13_{10}$

$10111_2 \rightarrow 23_{10}$

$$\begin{array}{r} 10_{10} \\ +\ 13_{10} \\ \hline 23_{10} \end{array}$$

$$\begin{array}{r} 10101 \\ +\ 11011 \\ +\ 10010 \\ 01100 \\ \hline 1001110_2 \end{array}$$

$10101_2 \rightarrow 21_{10}$

$11011_2 \rightarrow 27_{10}$

$10010_2 \rightarrow 18_{10}$

$01100_2 \rightarrow 12_{10}$

$1001110_2 \rightarrow 78_{10}$

$$\begin{array}{r} 21_{10} \\ 27_{10} \\ +\ 18_{10} \\ 12_{10} \\ \hline 78_{10} \end{array}$$

Resta binario y decimal

$$\begin{array}{r} 110101_2 \\ -\ 011011_2 \\ \hline 011010_2 \end{array}$$

$110101_2 \rightarrow 53_{10}$

$011011_2 \rightarrow 27_{10}$

$011010_2 \rightarrow 26_{10}$

$$\begin{array}{r} 53_{10} \\ 27_{10} \\ \hline 26_{10} \end{array}$$

$$\begin{array}{r} 1011100_2 \\ -\ 0100011_2 \\ \hline 0111001_2 \end{array}$$

$1011100_2 \rightarrow 92_{10}$

$0100011_2 \rightarrow 35_{10}$

$0111001_2 \rightarrow 57_{10}$

$$\begin{array}{r} 92_{10} \\ 35_{10} \\ \hline 57_{10} \end{array}$$

Multiplicacion binario y decimal

$$\begin{array}{r} 101_2 \\ *\ 100_2 \\ \hline 000 \\ 000 \\ 101 \\ \hline 10100_2 \end{array}$$

$101_2 \rightarrow 5_{10}$

$100_2 \rightarrow 4_{10}$

$10100_2 \rightarrow 10_{10}$

$$\begin{array}{r} 5_{10} \\ *\ 4_{10} \\ \hline 20_{10} \end{array}$$

$$\begin{array}{r} 110_2 \\ *\ 101_2 \\ \hline 110 \\ 000 \\ 110 \\ \hline 11110_2 \end{array}$$

$110_2 \rightarrow 6_{10}$

$101_2 \rightarrow 5_{10}$

$11110_2 \rightarrow 30_{10}$

$$\begin{array}{r} 6_{10} \\ 5_{10} \\ \hline 30_{10} \end{array}$$

Tema | Día | Mes | Año | ESTILO

- División binaria y decimal

```
 1 0 0 1 0 | 10
-10        |----
 0 0 0 1 0 | 1001
   -  1 0  |
   -------
     0 0
```

$10010_2 \rightarrow 18_{10}$

$10_2 \rightarrow 2_{10}$

$1001_2 \rightarrow 9_{10}$

```
 1 8 | 2
-1 8 | 9
 ----
 0 0
```

```
 1 0 1 0 0 | 100
-1 0 0     |-----
 0 0 1 0 0 | 101
   -  1 0 0|
   ---------
     (0 0 0)
```

$10100_2 \rightarrow 20_{10}$

$100_2 \rightarrow 4_{10}$

$101_2 \rightarrow 5_{10}$

```
2 0 | 4
2 0 | 5
 ---
 0 0
```

Notas

B. Ejercicios de Álgebra Booleana

Nombre: Andy Johel Mendoza Paíroga Fecha: 27/07/2025

Ejercicios de Álgebra Booleana:

- Simplificar expresiones usando mapas de Karnaugh.
- $F_1(A,B,C) = \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C} + AB\bar{C} + \bar{A}BC$

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

AB \ C	0	1
00	0	1
01	1	1
11	1	1
10	0	0

$F_{1min}(A,B,C) = B + \bar{A}C$

- $F_2(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C$

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

AB \ C	0	1
00	1	1
01	0	0
11	0	0
10	1	1

$F_{2min}(A,B,C) = \bar{B}$

- $F_3(A,B,C,D) = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}D + A\bar{B}CD + AB\bar{C}D + ABCD$

A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

AB \ CD	00	01	11	10
00	1	0	0	1
01	1	0	0	0
11	0	1	1	0
10	0	1	1	0

$F_{3min}(A,B,C,D) = AD + \bar{A}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{D}$

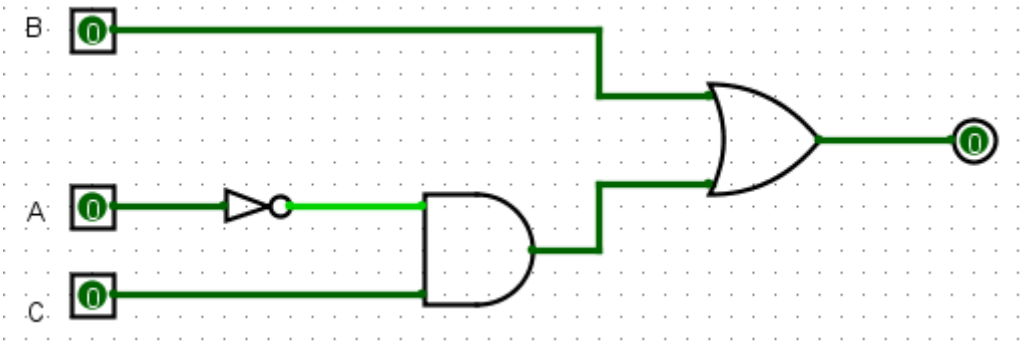
Diseñar y probar circuitos lógicos básicos utilizando software de simulación.

Para realizar la simulación de los circuitos lógicos, se empleó el software Open Source "Logisim Evolution", y se diseñó los circuitos lógicos con el resultado de la simplificación de expresiones usando mapas de Karnaugh del ejercicio anterior.

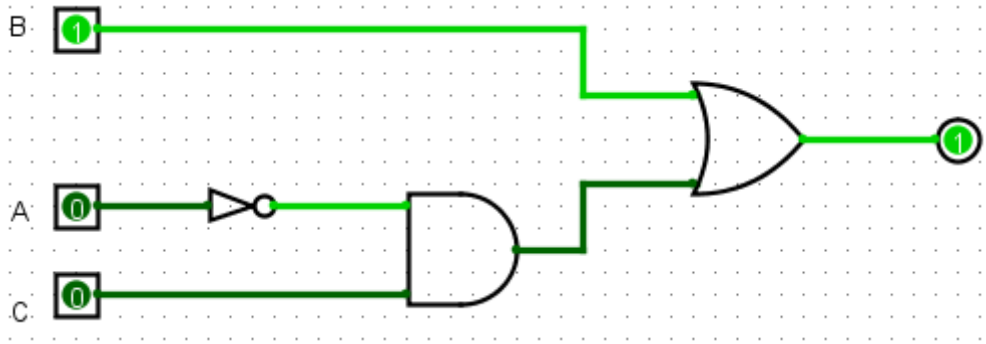
- **$F_1(A, B, C) = \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C} + AB\bar{C} + \bar{A}BC$**

El resultado de la simplificación es: $B + \bar{A}C$, lo que nos quiere decir que la salida será alta cuando B sea de nivel alto o cuando A sea de nivel bajo y C sea de nivel alto.

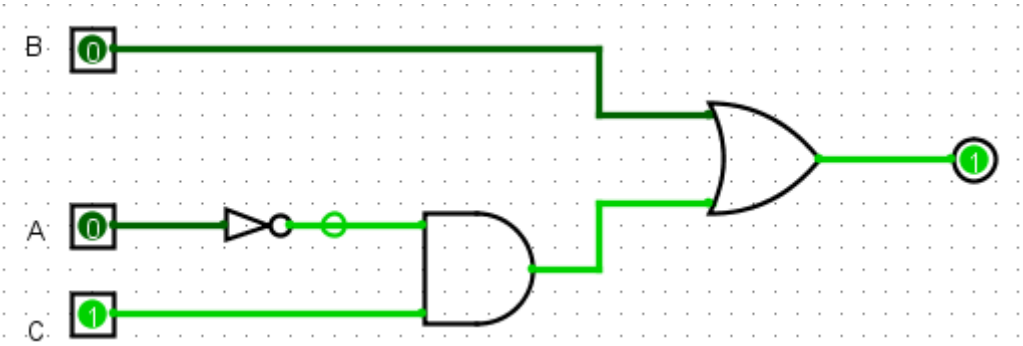
Circuito:



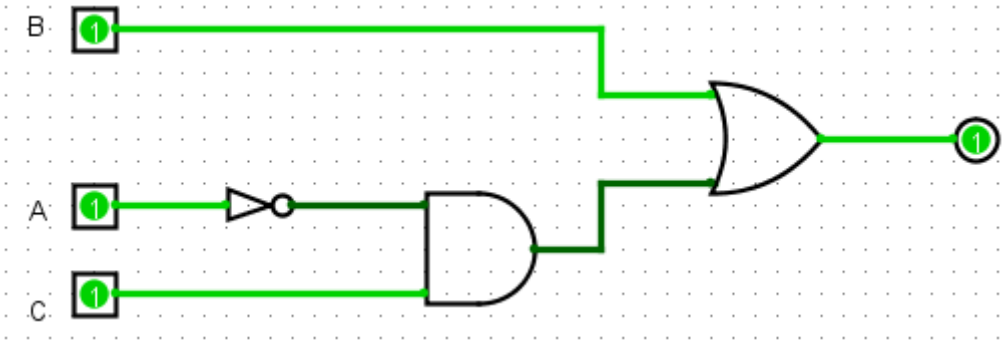
Cuando B es de nivel alto:



Cuando A es de nivel bajo y C es de nivel alto:



Cuando A, B y C son de nivel alto:

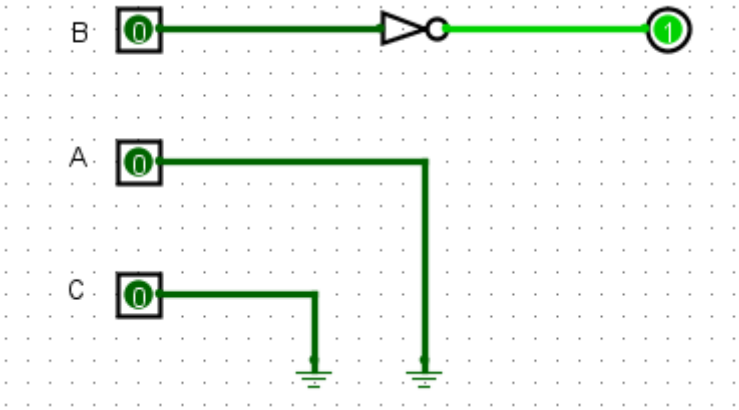


Todo esto comprueba de que si se cumple lo que dice la expresión simplificada.

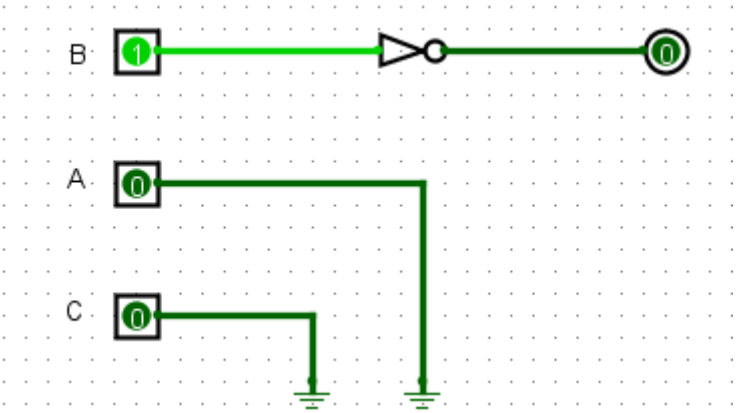
- **$F_2(A, B, C) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C$**

El resultado de la simplificación es: $\bar{B}$, lo que quiere decir que la salida será de nivel alto únicamente cuando B sea de nivel bajo (negada), y en el circuito como A y C no inciden en el resultado de salida se las manda a tierra.

Circuito:



Cuando B es de nivel alto:

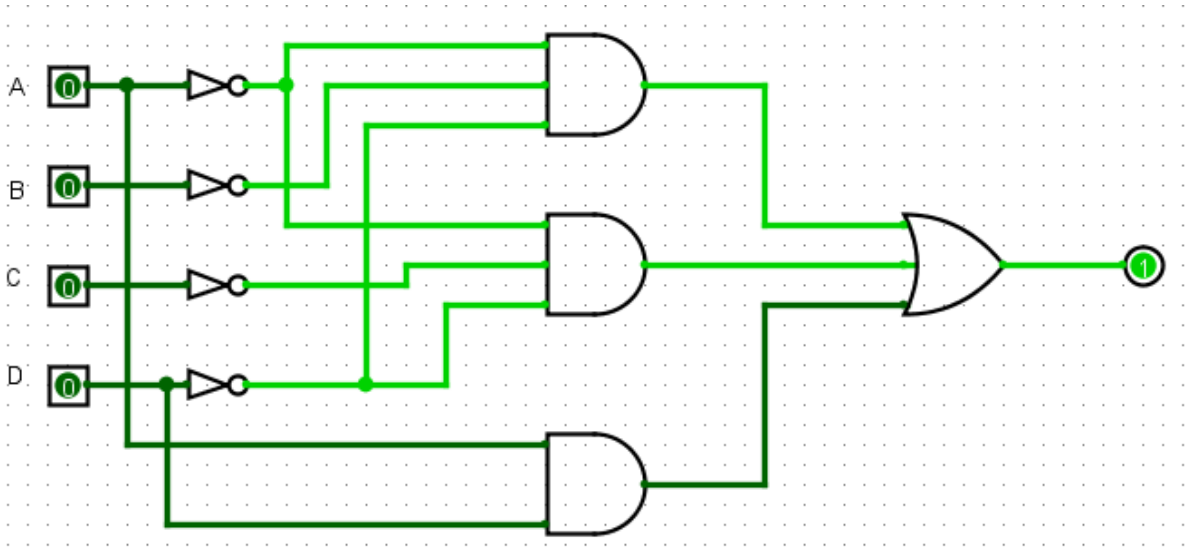


Esto refleja que si se cumple lo que está en la expresión simplificada

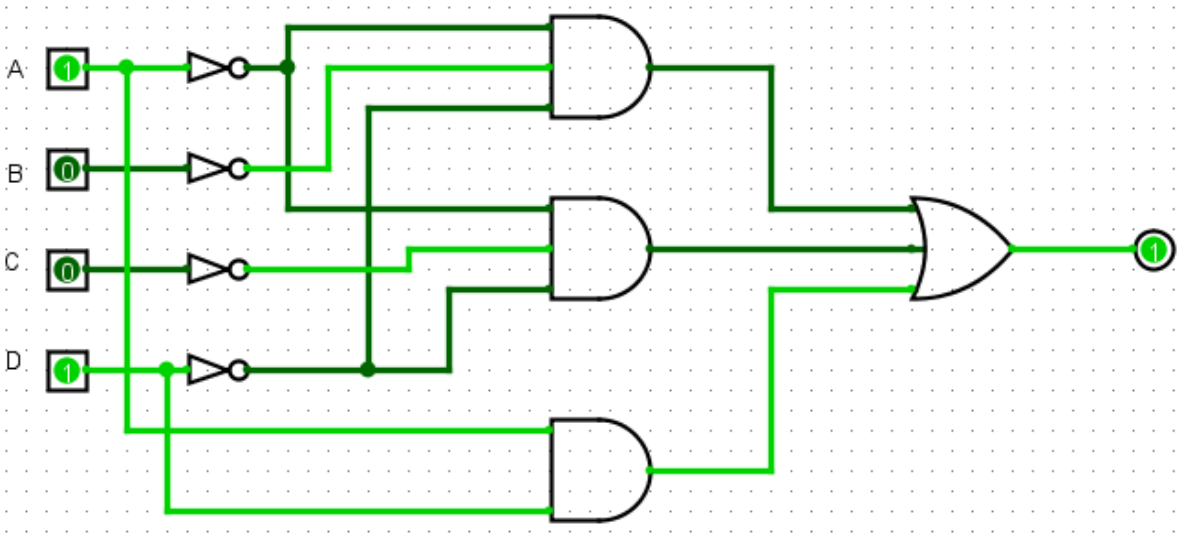
- **$F_3(A, B, C, D) = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}D + A\bar{B}CD + AB\bar{C}D + ABCD$**

El resultado de la simplificación es: $AD + \bar{A}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{D}$, lo que quiere decir que la salida será de nivel alto cuando A y D sean de nivel alto, o A, C y D sean de nivel bajo (negadas), o cuando A, B y D sean de nivel bajo (negadas).

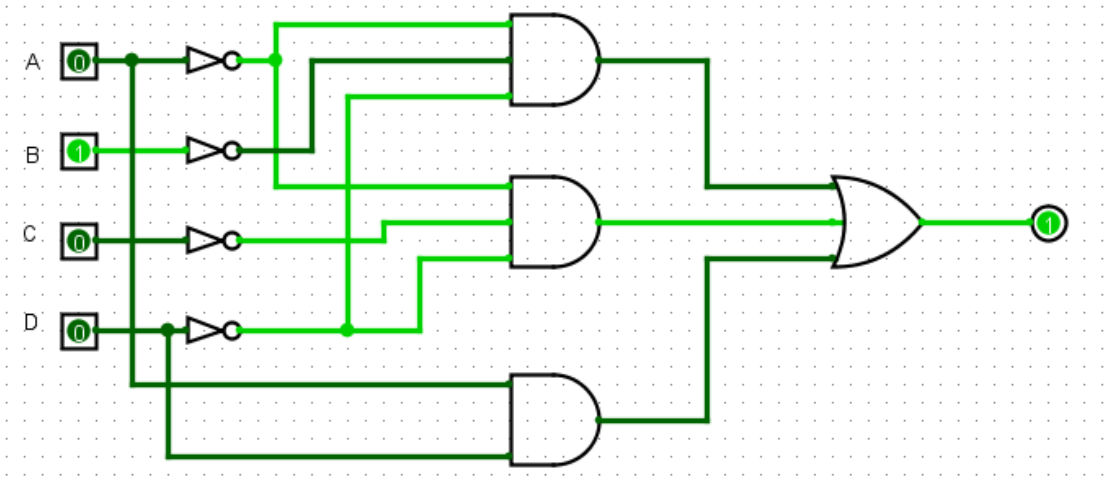
Circuito:



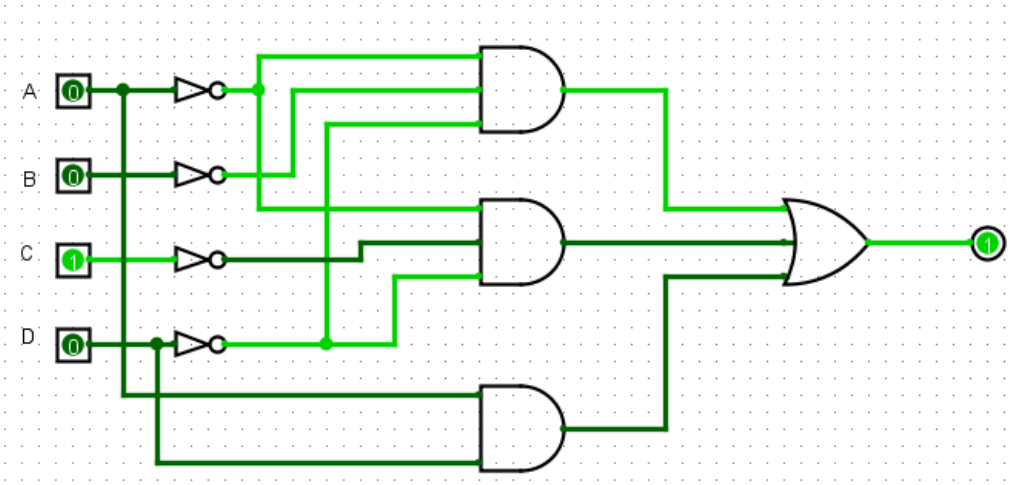
Cuando A y D son de nivel alto:



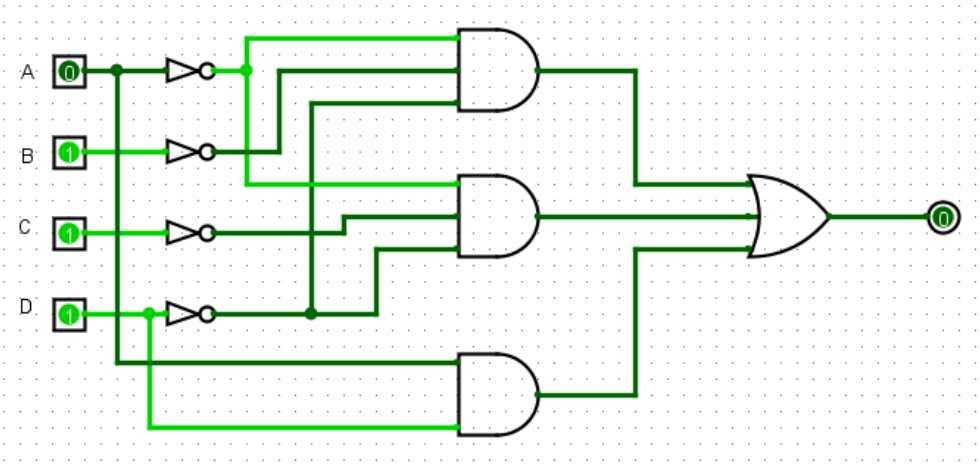
Cuando A, C y D son de nivel bajo (negadas):



Cuando A, B y D son de nivel bajo (negadas):



Cuando A es de nivel bajo y las demás son de nivel alto:



Todo esto comprueba de que la expresión simplificada es correcta.

C. Diseño de Circuitos Combinacionales

Ronaldo Zamora 27/07/2025

Diseño de Circuitos Combinacionales:

- Construcción de un sumador binario completo.

A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Cout

A \ BCin	00	01	11	10
0	0	0	1	0
1	0	1	1	1

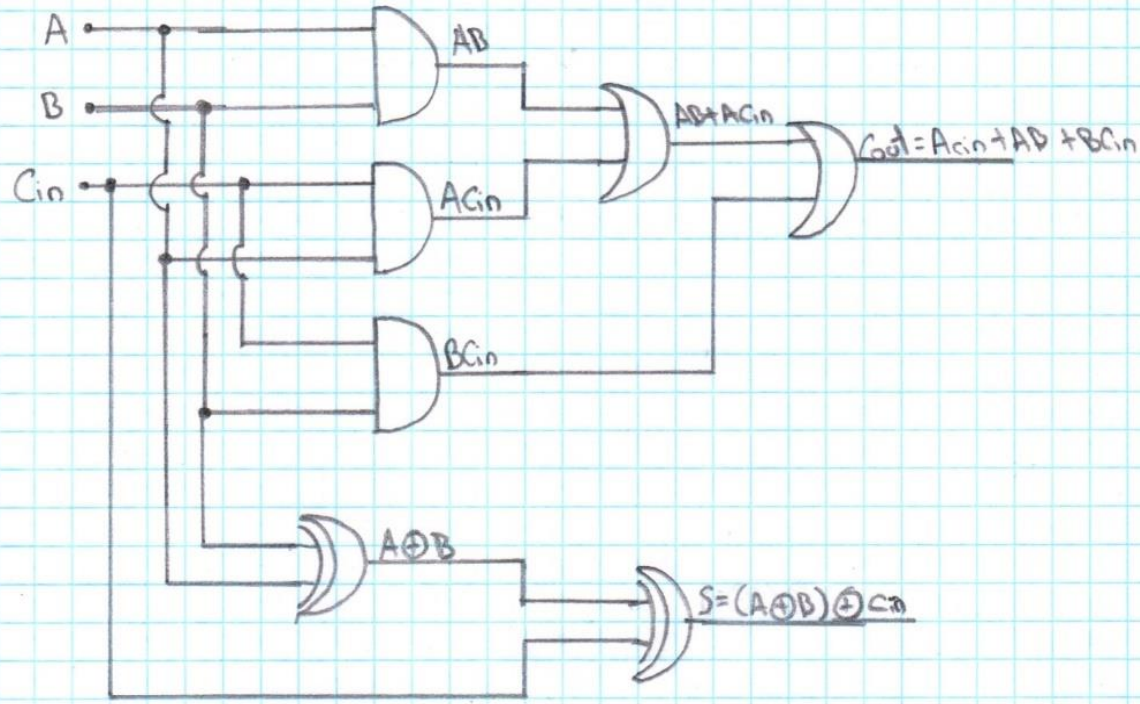
$Cout = ACin + AB + BCin$

S

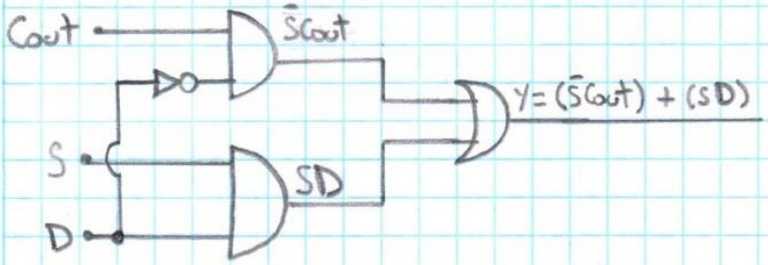
A \ BCin	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$S = A\bar{B}\bar{C}in + \bar{A}B\bar{C}in + ABCin + \bar{A}\bar{B}Cin$

$S = (A \oplus B) \oplus Cin$



- Implementar un multiplexor simple para controlar el flujo de datos



D. Estudio de Microprocesadores y Microcontroladores

Arquitectura de un microcontrolador y microprocesador

Los microprocesadores y microcontroladores son fundamentales en el desarrollo de tecnologías inteligentes, y su arquitectura ha evolucionado para adaptarse a las necesidades del Internet de las Cosas (IoT) y la inteligencia artificial (IA). El microcontrolador, según el artículo ″A Smart Microcontroller Architecture for the Internet of Things″, es un chip que combina procesador, memoria y periféricos, ideal para tareas específicas y de bajo consumo. La arquitectura propuesta incluye capacidades de inteligencia local como lógica difusa y redes neuronales simples, lo que permite tomar decisiones sin depender de la nube. Además, se plantea una red de microcontroladores conectados mediante un bus, que colaboran entre sí, mejorando la eficiencia y escalabilidad en aplicaciones como sensores o cerraduras inteligentes. Por su parte, el artículo *"Advancements in Microprocessor Architecture for Ubiquitous AI"* explica cómo los microprocesadores han pasado de tareas generales a procesar IA avanzada, incorporando núcleos múltiples, GPUs, TPUs y conjuntos de instrucciones especializados. Estas mejoras permiten realizar tareas como visión por computadora o reconocimiento de voz en tiempo real. Sin embargo, también enfrentan desafíos como el manejo del calor, el consumo energético y la compatibilidad de software. En conjunto, ambas arquitecturas trabajan de forma complementaria: los microcontroladores procesan tareas simples y locales, mientras los microprocesadores se encargan de operaciones más complejas, logrando sistemas más inteligentes y eficientes [35], [36].

Los avances en la arquitectura de microprocesadores y microcontroladores reflejan el crecimiento de tecnologías como el IoT y la inteligencia artificial. Los microcontroladores, según la propuesta del artículo sobre IoT, integran en un solo chip elementos esenciales para controlar dispositivos de forma autónoma y eficiente. Se destacan por su bajo consumo, capacidad de tomar decisiones simples con lógica programable y de formar redes con otros microcontroladores para resolver tareas distribuidas. Esto los hace ideales para entornos donde la respuesta rápida y local es crucial. En cambio, los microprocesadores, como detalla el artículo sobre IA ubicua, han evolucionado para soportar cargas de trabajo mucho más pesadas, gracias a múltiples núcleos, aceleradores y arquitecturas optimizadas para aprendizaje profundo, visión artificial y procesamiento de datos complejos. Si bien ofrecen alto rendimiento, también presentan retos como el calor generado y la complejidad del software. Ambos tipos de arquitectura se complementan: mientras los microcontroladores ejecutan

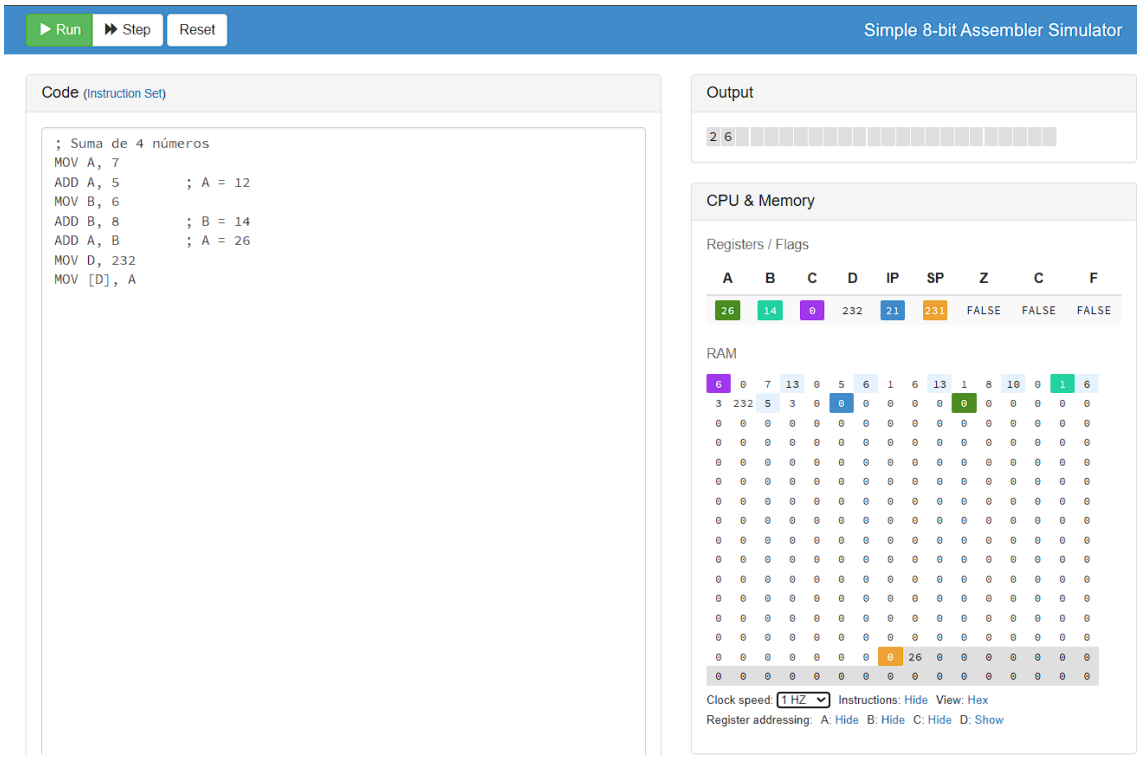
acciones locales y específicas, los microprocesadores manejan análisis profundos y tareas globales, formando así la base de sistemas inteligentes, conectados y eficientes [36].

Analizar el ciclo de instrucciones y la velocidad del microprocesador con ejemplos prácticos.

Para realizar este análisis se hicieron simulaciones prácticas en un simulador online de arquitectura de computadoras denominado “Simple 8-bit Assembler Simulator”, que emula una arquitectura básica similar a la del Intel 8086.

Para el análisis se codificó en lenguaje ensamblador utilizando instrucciones como MOV, ADD, entre otras, que permitieron observar el funcionamiento básico del ciclo de instrucciones y el cambio en los registros durante la ejecución, con el objetivo de determinar la velocidad del microprocesador en distintas velocidades de reloj como 1Hz, 4Hz, 8Hz y 16Hz.

Primer ejemplo: Suma de 4 números

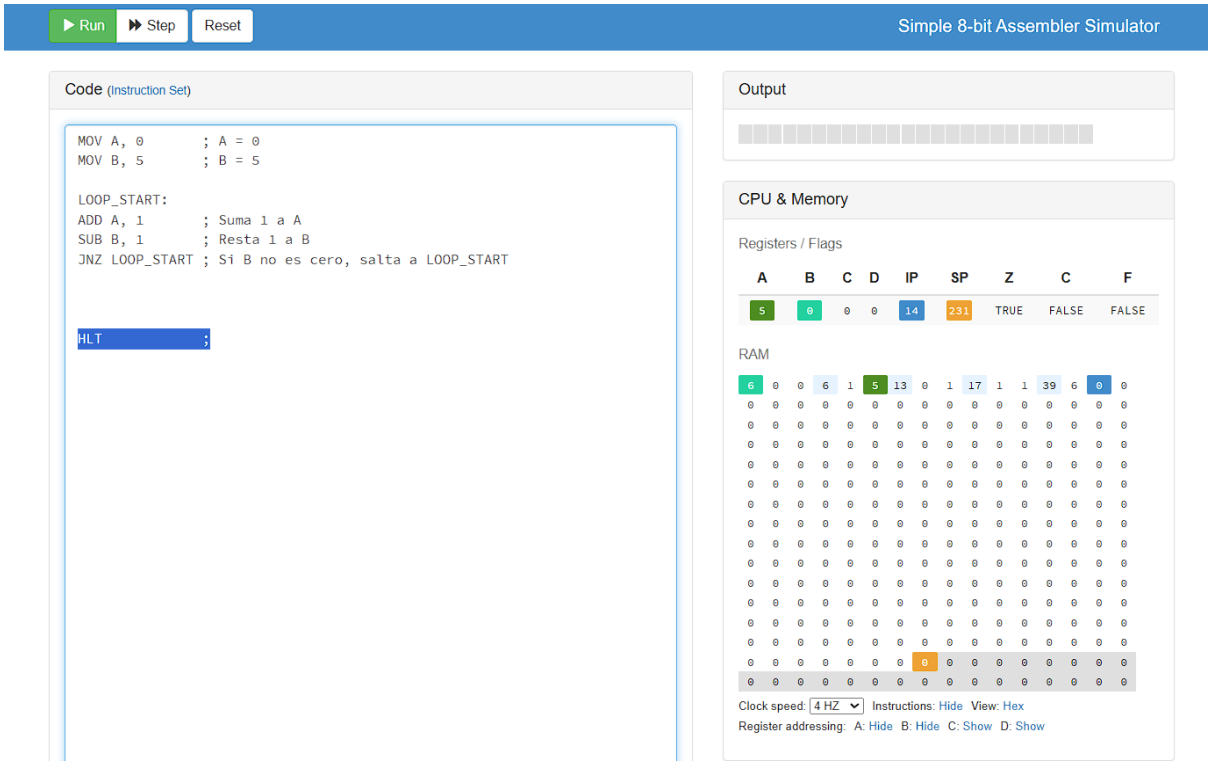


En este ejemplo existen 7 ciclos de instrucción, ya que en ensamblador cada instrucción ejecutable se considera un ciclo de instrucción.

Al evaluar la ejecución a 1Hz, el tiempo que le tomó es de 7 segundos, ya que 1Hz equivale a una instrucción por segundo. A una frecuencia de 4 Hz, el tiempo de ejecución fue de 1.75

segundos. Al evaluarlo en 8 Hz, la ejecución tardó en finalizar en 875 milisegundos. En 16 Hz le tomó un tiempo de 437.5 milisegundos. Y si lo evaluamos con la frecuencia promedio de un microprocesador de la actualidad que va a 3.7 GHz, solo le tomaría 1.8919 nanosegundos.

Ejemplo 2: Suma 1 en el registro A hasta que B sea 0



Este ejemplo cuenta con 2 ciclos de instrucción iniciales, 3 en el loop que multiplicado por 5 iteraciones serían 15 ciclos de instrucciones y 1 en el final, en total 18 ciclos de instrucciones.

Al evaluar la ejecución a una frecuencia de 1Hz le tomó 18 segundos. A una frecuencia de 4 Hz tardó 4.5 segundos. Con una frecuencia de 8 Hz demoró 2.25 segundos en finalizar la ejecución. Y a una frecuencia de 16 Hz solo le tomó 1.125 segundos en finalizar. Si lo evaluamos a una frecuencia promedio de un microprocesador de la actualidad que va a 3.7 GHz, sería prácticamente imperceptible tomándole solo 4.87 nanosegundos aproximadamente.

CONCLUSIÓN

El estudio de la aritmética de la computadora y el álgebra de Boole permite comprender los fundamentos que sustentan el funcionamiento de los sistemas digitales modernos. La representación binaria, junto con las operaciones lógicas y aritméticas, es esencial para el procesamiento de datos en computadoras. El uso del álgebra booleana en el diseño de circuitos lógicos facilita la optimización y simplificación de procesos computacionales. Asimismo, el análisis de microprocesadores y microcontroladores evidencia cómo estos dispositivos ejecutan tareas complejas o específicas con eficiencia, siendo pilares fundamentales en tecnologías actuales como el Internet de las Cosas y la inteligencia artificial.

BIBLIOGRAFÍA

[1] E. E. Swartzlander, "Computer Arithmetic", en *Computer Arithmetic*, WORLD SCIENTIFIC, 2015, pp. 1–398. doi: 10.1142/9789814651578.

[2] A. Lloris Ruiz, E. Castillo Morales, L. Parrilla Roure, A. García Ríos, y M. J. Lloris Meseguer, "Number Systems", 2021, pp. 1–75. doi: 10.1007/978-3-030-67266-9_1.

[3] B. J. LaMeres, "Floating-Point Systems", en *Introduction to Logic Circuits & Logic Design with Verilog*, Cham: Springer International Publishing, 2024, pp. 465–514. doi: 10.1007/978-3-031-43946-9_14.

[4] L. Strickland y H. R. Lewis, *Leibniz on Binary*. The MIT Press, 2022. doi: 10.7551/mitpress/14123.001.0001.

[5] I. Spiridonov, "Arithmetic of binary equivalents of decimal numbers", el 23 de noviembre de 2022. doi: 10.36227/techrxiv.19294511.v4.

[6] R. Mian, M. Shintani, y M. Inoue, "Hardware–Software Co-Design for Decimal Multiplication", *Computers*, vol. 10, núm. 2, p. 17, ene. 2021, doi: 10.3390/computers10020017.

[7] L.-K. Wang, M. J. Schulte, J. D. Thompson, y N. Jairam, "Hardware Designs for Decimal Floating-Point Addition and Related Operations", *IEEE Transactions on Computers*, vol. 58, núm. 3, pp. 322–335, mar. 2009, doi: 10.1109/TC.2008.147.

[8] Á. Pereda Loriente, J. A. González-Calero, S. Tirado-Olivares, y J. del Olmo-Muñoz, "Enhancing mathematics performance in primary education: The impact of personalized

learning on fractions and decimal numbers", *Educ Inf Technol (Dordr)*, feb. 2025, doi: 10.1007/s10639-025-13428-5.

[9] S. Kurgalin y S. Borzunov, "Boolean Algebra", 2020, pp. 217–249. doi: 10.1007/978-3-030-42221-9_6.

[10] K. Erciyes, "Boolean Algebras and Combinational Circuits", 2021, pp. 173–195. doi: 10.1007/978-3-030-61115-6_9.

[11] R. Drechsler y S. Huhn, Eds., *Advanced Boolean Techniques*. Cham: Springer International Publishing, 2023. doi: 10.1007/978-3-031-28916-3.

[12] S. Huhn y R. Drechsler, *Design for Testability, Debug and Reliability*. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-69209-4.

[13] D. M. Miller y G. W. Dueck, "Translation Techniques for Reversible Circuit Synthesis with Positive and Negative Controls", en *Recent Findings in Boolean Techniques*, Cham: Springer International Publishing, 2021, pp. 143–165. doi: 10.1007/978-3-030-68071-8_7.

[14] A. Mahzoon, D. Große, y R. Drechsler, *Formal Verification of Structurally Complex Multipliers*. Cham: Springer International Publishing, 2023. doi: 10.1007/978-3-031-24571-8.

[15] M. Garrido, "Simplifying Karnaugh Maps by Making Groups of Non-power-of-two Elements", *Circuits Syst Signal Process*, vol. 41, núm. 10, pp. 5895–5902, oct. 2022, doi: 10.1007/s00034-022-02040-4.

[16] N. I. Georgiev, V. V. Bakov, y V. B. Bojinov, "A Tutorial Review on the Fluorescent Probes as a Molecular Logic Circuit—Digital Comparator", *Molecules*, vol. 28, núm. 17, p. 6327, ago. 2023, doi: 10.3390/molecules28176327.

[17] S. S. Srikant y P. K. Chaturvedi, *Basic Electronics Engineering*, 1a ed. Singapore: Springer Singapore, 2020. doi: 10.1007/978-981-13-7414-2.

[18] J. F. Groote, R. Morel, J. Schmaltz, y A. Watkins, *Logic Gates, Circuits, Processors, Compilers and Computers*. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-68553-9.

[19] Neha M Harapanhalli, Kavipriya M, Ishwari Jigajinni, y Dr. Keerti Kulkarni, “Implementation of Combinational and Sequential Logic Circuits using Quantum Computing”, *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 430–439, jun. 2023, doi: 10.32628/CSEIT23903106.

[20] S. Bandyopadhyay, “Nanomagnetic Boolean Logic—The Tempered (and Realistic) Vision”, *IEEE Access*, vol. 9, pp. 7743–7750, 2021, doi: 10.1109/ACCESS.2021.3049333.

[21] S.Nagaraj, G. M. Sreerama Reddy, y S. Aruna Mastani, “Design and Analysis of Novel Full Adder using ECRAAL”, *International Journal of Computational and Experimental Science and Engineering*, vol. 11, núm. 1, ene. 2025, doi: 10.22399/ijcesen.848.

[22] A. Lloris Ruiz, E. Castillo Morales, L. Parrilla Roure, A. García Ríos, y M. J. Lloris Meseguer, *Arithmetic and Algebraic Circuits*, vol. 201. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-67266-9.

[23] M. Sanadhya y D. K. Sharma, “Design and implementation of full subtractor using different adiabatic techniques”, en *2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*, IEEE, dic. 2020, pp. 102–106. doi: 10.1109/WIECON-ECE52138.2020.9397967.

[24] M. Bansal, H. Singh, y G. Sharma, “A Taxonomical Review of Multiplexer Designs for Electronic Circuits & Devices”, *Journal of Electronics and Informatics*, vol. 3, núm. 2, pp. 77–88, abr. 2021, doi: 10.36548/jei.2021.2.001.

[25] A. Al Zaman y N. J. Monira, “AN OVERVIEW OF MICROPROCESSORS AND ASSEMBLY LANGUAGE PROGRAMMING ”, *Advances in Interconnect Technologies: An International Journal (AITIJ)* , vol. 1, núm. oct, sep. 2022, doi: 10.5281/zenodo.7081411.

[26] H. Z. Abdullayevich, “History, Structure And Types Of Microprocessors”, *The American Journal of Interdisciplinary Innovations and Research*, vol. 02, núm. 11, pp. 39–46, nov. 2020, doi: 10.37547/tajiir/Volume02Issue11-08.

[27] A. A. Antonov y A. A. Krasnyuk, “The internal structure of microprocessors for industrial control and data processing systems”, *IOP Conf Ser Mater Sci Eng*, vol. 1061, núm. 1, p. 012003, feb. 2021, doi: 10.1088/1757-899X/1061/1/012003.

[28] D. Suárez, F. Almeida, y V. Blanco, “Comprehensive analysis of energy efficiency and performance of ARM and RISC-V SoCs”, *J Supercomput*, vol. 80, núm. 9, pp. 12771–12789, jun. 2024, doi: 10.1007/s11227-024-05946-9.

[29] G. Nikolic, B. Dimitrijevic, T. Nikolic, y M. Stojcev, “Fifty years of microprocessor evolution: from single CPU to multicore and manycore systems”, *Facta universitatis - series: Electronics and Energetics*, vol. 35, núm. 2, pp. 155–186, 2022, doi: 10.2298/FUEE2202155N.

[30] C. Ünsalan, H. D. Gürhan, y M. E. Yücel, *Embedded System Design with ARM Cortex-M Microcontrollers*. Cham: Springer International Publishing, 2022. doi: 10.1007/978-3-030-88439-0.

[31] C. Ünsalan, B. Höke, y E. Atmaca, *Embedded Machine Learning with Microcontrollers*. Cham: Springer International Publishing, 2025. doi: 10.1007/978-3-031-69421-9.

[32] E. H. Currie, “Microcontroller Subsystems”, en *Mixed-Signal Embedded Systems Design*, Cham: Springer International Publishing, 2021, pp. 35–97. doi: 10.1007/978-3-030-70312-7_2.

[33] B. Goossens, *Guide to Computer Processor Architecture*. Cham: Springer International Publishing, 2023. doi: 10.1007/978-3-031-18023-1.

[34] P. Bulić, *Understanding Computer Organization*. Cham: Springer International Publishing, 2024. doi: 10.1007/978-3-031-58075-8.

[35] Z. Wu, K. Qiu, y J. Zhang, “A Smart Microcontroller Architecture for the Internet of Things”, *Sensors*, vol. 20, núm. 7, p. 1821, mar. 2020, doi: 10.3390/s20071821.

[36] F. H. Khan, M. A. Pasha, y S. Masud, “Advancements in Microprocessor Architecture for Ubiquitous AI—An Overview on History, Evolution, and Upcoming Challenges in AI Implementation”, *Micromachines (Basel)*, vol. 12, núm. 6, p. 665, jun. 2021, doi: 10.3390/mi12060665.

ANEXO

Link del repositorio en GitHub: https://github.com/AndyMendoza0308/ArqComp-Grupo-F