UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO
FACULTAD CIENCIAS DE LA INGENIERÍA
CARRERA SOFTWARE

INTEGRANTES:
CORONEL TOALA CRISTHIAN ALEXIS – ccoronelt2@uteq.edu.ec
LLERENA ABRIL ANGELINA JULEXY – allerenaa@uteq.edu.ec
NARANJO FLORES ANDERSON JEAMPIERE – anaranjof2@uteq.edu.ec
NIVELA SAAVEDRA JOSE RICARDO – jnivelas2@uteq.edu.ec

DOCENTE:
PHD. GUERRERO ULLOA GLEISTON CICERON

CURSO Y PARALELO:
SEGUNDO SOFTWARE “B”

GRUPO:
D

ASIGNATURA:
ARQUITECTURA DE COMPUTADORAS

TEMA:
Aritmética de la Computadora y Álgebra Boole

PERIODO ACADÉMICO:
SPA 2025 – 2026

ÍNDICE

FUNDAMENTO TEÓRICO

1. ARITMÉTICA BINARIA Y DECIMAL

1.1. Sistema Binario

El sistema binario más conocido como lenguaje maquina se viene desarrollando hace muchos años atrás, llego a ser lo que es ahora gracias a el aporte de grandes mentes diferentes [1].

El sistema y el código binario están estrechamente relacionados, pero siguen siendo cosas diferentes el código binario es el lenguaje al cual se traducen las instrucciones de los programas informáticos y el sistema binario es un sistema numérico al igual que el sistema decimal [2].

Al ser un sistema numérico este se puede operar aritméticamente, sumas, restas, multiplicaciones y divisiones, pero también se usa para representar datos ya que gracias a códigos como el ASSCI se puede usar el código binario para representar cada letra del abecedario incluso colores que suelen estar codificados en binario.

1.1.1. Suma binaria

La suma binaria es exactamente igual a la suma decimal incluso haciendo uso de los acarreos, claro sin tomar en cuenta que en esta solo existen 2 números [3]. La diferencia es que en decimal los números llegan hasta el 9 y en binario solo hasta el 1 así que algo en decimal tan simple como un 1+1 es diferente en binario.

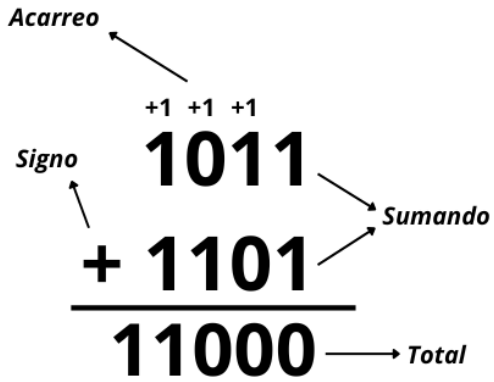


Figura 1. Suma binaria

1.1.2. Resta de binarios

Existen 2 formas de hacerlo la que es parecida al método de los decimales en donde se resta y se realizan prestamos en caso de que el digito del sustraendo sea mayor que el del minuendo y la otra que es en lenguaje máquina, ya que las máquinas para restar suman un complemento a él sustraendo y luego lo suman con el minuendo ya que debido a la estructura de la computadora esto agiliza los procesos [2].

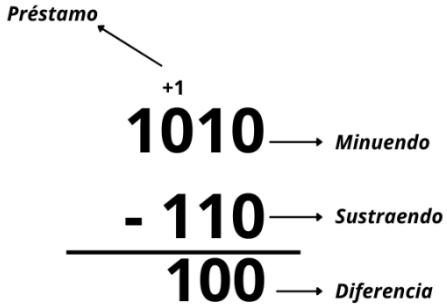


Figura 2. Resta Binaria

La resta que se hace en la maquina consiste en invertir el sustraendo y sumarle 1, luego de esto se suma con el minuendo y si en la respuesta sobra un acarreo este se ignora y ahí daría la respuesta [2]

1.1.3. Multiplicación de binarios

Para hacer la multiplicación de binarios es la siguiente tabla con las reglas básicas:

Tabla 1. Reglas básicas de la multiplicación binaria

*	0	1
0	0	0
1	0	1

La *Tabla 1* dice que 0*0 = 0, 0*1 = 0, 1*0 = 0, 1*1= 1, algo parecido al sistema decimal, mediante estas reglas podremos desarrollar la multiplicación de binarios [4].

Visto de otra manera lo que se hace en una, multiplicación de binarios es que si el multiplicador es 1 repetir el numero binario y si es 0 poner tantos 0 como dígitos haya en el número binario, por cada multiplicación la siguiente tiene que ir un digito desplazado hacia la izquierda, después de multiplicar todos los dígitos del multiplicador se suman los resultados [3]

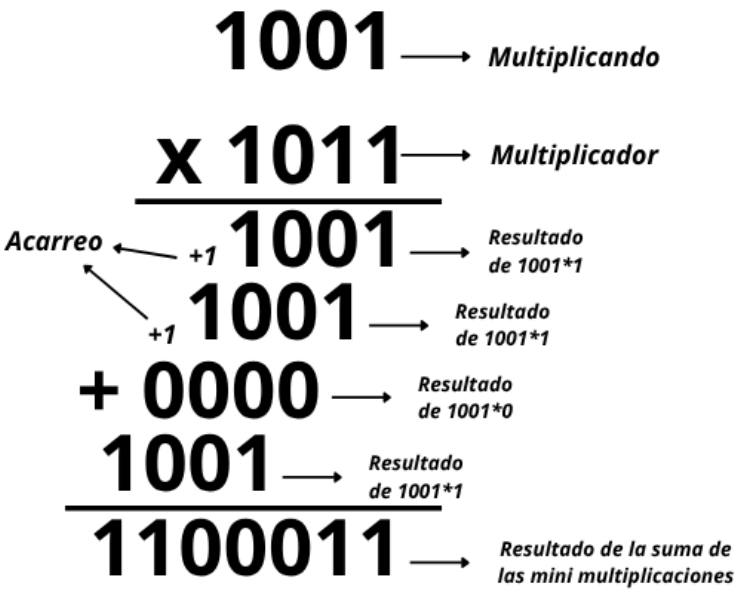


Figura 3. Multiplicación Binaria

1.1.4. División de binarios

En binario la división es demasiado parecido a la división en un sistema decimal, eso quiere decir que en la división binaria también se usa una resta iterada [3].

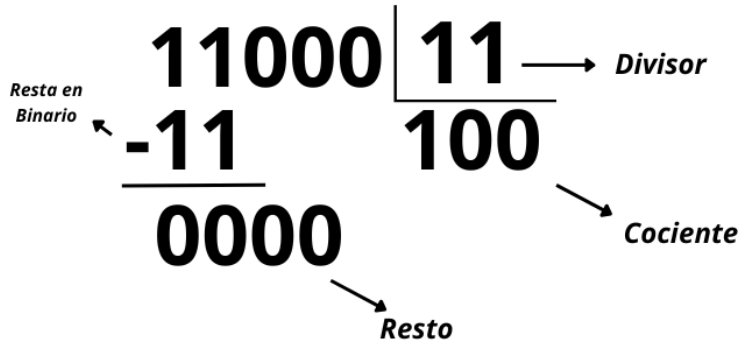


Figura 4. División Binaria

Para convertir un numero en base binario a base 10 lo que hay que hacer es colocar cada 0 y 1 en el orden en que están, colocar un dos debajo o encima de cada uno y elevar cada dos empezando en 0 hasta n número, por ejemplo, si el numero binario fuera 110011 el 2 empezaría con exponente 0 y terminaría en exponente 5 [2]

Luego de eso elevamos cada 2 y cada respuesta se la multiplica por el digito asignado, siguiendo con el ejemplo pasado empezaría con 2^0 *1 la respuesta seria 1 por leyes de la potenciación, así sucesivamente [2].

Los resultados de estas multiplicaciones serán sumados y obtendremos el valor en decimal [2]

2. ÁLGEBRA DE CIRCUITOS DIGITALES Y FUNCIÓN BOOLE

El proceso de simplificación de las funciones booleanas es un estudio básico y esencial para conseguir sistemas que satisfagan unas consideraciones de eficiencia, velocidad o consumo energético, y fotograma a fotograma cada bit cuenta. Mediante el álgebra booleana y su aplicación a las compuertas lógicas los ingenieros son capaces de pasar de estructuras complejas a estructuras mínimas, optimizando así recursos y costes. En el ámbito educativo han ido surgiendo distintas metodologías que pretenden facilitar este proceso, convirtiéndose así desde manuales muy teóricos a herramientas digitales que favorecen el aprendizaje activo. En este documento van a ser analizadas cinco fuentes clave que van desde los fundamentos teóricos hasta las modernas propuestas pedagógicas que se basan en la gamificación o la visualización.

2.1. Fundamentos Lógicos y Compuertas Digitales

Elahi et al. (2022), han presentado un enfoque exhaustivo sobre los principios mediante los cuales la lógica booleana se basa en el uso de compuertas lógicas como bloques de la construcción para circuitos digitales [5]. En este sentido, el trabajo se centra en compuertas AND, OR, NOT, NAND, NOR, XOR y XNOR, y profundiza en la teoría de leyes algebraicas como la conmutativa, la distributiva y la ley de Morgan. También hace un primer acercamiento a las clasificaciones de circuitos integrados (SSI, MSI, LSI y VLSI) para el propio diseño escalable de sistemas embebidos, de ordenadores y microcontroladores. Esto, como tipo de contenido, se puede considerar la base teórica que da sustento al análisis lógico y posterior simplificación de expresiones booleanas mediante su utilización en aplicaciones reales.

2.2. MiniBool como Herramienta Ramificada para la Enseñanza

Jiménez-Hernández et al. (2020), nos presentan una alternativa innovadora de enseñanza, MiniBool, una herramienta ramificada para interactuar con funciones booleanas por medio de mapas de Karnaugh y ejercicios propios, y en contraposición con el habitual lápiz y papel, MiniBool tiene la ventaja de dar feedback inmediato, niveles progresivos de dificultad y una

interfaz amigable que facilita la motivación y el aprendizaje significativo; la gamificación permite al estudiante estar inmerso en procesos de ensayos que ayudan a trabajar la memoria operativa y el razonamiento lógico. Este recurso no solo destierra la explicación técnica de los conceptos sino que incluye mecánicas de modalidad de aprendizaje de corte moderno en el aula virtual [6].

cd\ab	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	2	6	14	10
10	3	7	15	11

abcd	
0000: 0	1000: 8
0001: 1	1001: 9
0010: 2	1010: 10
0011: 3	1011: 11
0100: 4	1100: 12
0101: 5	1101: 13
0110: 6	1110: 14
0111: 7	1111: 15

Figura 5. Mapa de Karnaugh

2.3. Evaluación Comparativa de Software Educativo

Camacho-Mendoza et al. (2021), llevaron a cabo un estudio exhaustivo sobre el efecto de distintas plataformas del ámbito de la simplificación lógica en estudiantes de ingeniería [7]. Realizan una comparación de tres herramientas disponibles como son, MiniBool, BooleanSoft y Karnaugh Analyzer, haciendo un repaso y comparación sobre características como usabilidad, exactitud en resultados y retención de aprendizaje. A partir del análisis, aunque todas las plataformas analizadas presentan para los alumnos ventajas evidentes, MiniBool es la más aceptada por su diseño interactivo y por la posibilidad de personalizar la enseñanza a los estilos cognitivos de cada alumno. El estudio incluye métricas estadísticas: tasas de aciertos, velocidad de resolución de problemas, o la percepción del alumno acerca de su progreso, lo cual da cierto rigor científico al uso del software especializado como soporte pedagógico.

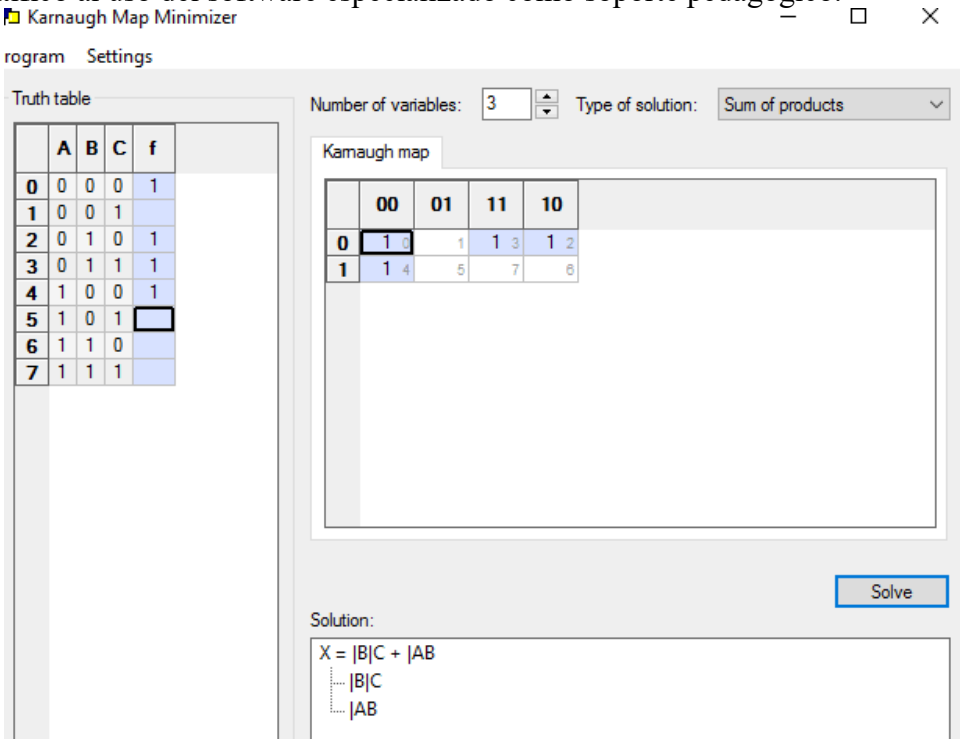


Figura 6. Herramientas TIC

2.4. Animación como Técnica de Visualización de Conceptos

Carpinelli (2023), expone un modelo educativo apoyado en animaciones, el cual hace posible que se pueda imaginar el comportamiento de compuertas, de combinaciones lógicas y de diagramas binarios de manera gráfica [8]. Esto es especialmente útil para los estudiantes que tienen un aprendizaje más visual, porque estos conceptos abstractos de la lógica digital los puede traducir en secuencias animadas donde se ven las distintas transiciones, las operaciones y los resultados esperados. La guía tiene también una utilidad para los educadores que pretenden innovar sus estrategias didácticas porque ofrece ejemplos intuitivos que pueden ser usados en clase o como material para el autoaprendizaje. La animación transforma simplificadamente las barreras cognitivas típicas que surgen en los temas altamente técnicos.

2.5. Minimización Gráfica con Mapas de Karnaugh

Brock J. LaMeres (2019), presenta una exposición detallada del uso de los mapas de Karnaugh como herramienta de minimización y optimización gráfica de funciones booleanas [9]. La publicación aborda los K-maps para funciones de dos, tres y cuatro variables, incluyendo todo, desde que cada celda del mapa contiene una combinación de entrada, hasta que las agrupaciones por vecindad de un bit permiten simplificar la lógica sin manipulación algebraica compleja. No se olvida incluir también el "envolvimiento" por bordes del mapa, que es clave para poder reconocer agrupaciones en mapas más amplios. Este tipo de tratamiento práctico es altamente complementario del análisis lógico convencional, ya que permite entender cómo detectar redundancias y descomponer expresiones en su forma óptima gracias a patrones visuales.

3. CIRCUITOS LÓGICOS Y SISTEMAS DIGITALES

Tal como mencionan Lin et al. (2022), la lógica digital y los circuitos digitales son el fundamento de todas las aplicaciones electrónicas y permiten la representación y manipulación fiable de la información binaria; la evolución de los circuitos lógicos y de los sistemas digitales se ha dado gracias a la integración de modelos basados en algorítmica, de automatización y de materiales emergentes, como consecuencia de lo cual se han conseguido circuitos digitales más pequeños, más rápidos y con un mejor consumo energético [10].

La cada vez mayor demanda de dispositivos inteligentes y conectados que Roy et al. (2021), mencionan lleva a que cada vez se alcancen niveles de complejidad mayores que se traducen en la exigencia de mejoras constantes en el diseño y la funcionalidad de tales sistemas [11].

3.1. Circuitos Lógicos Fundamentales

Los circuitos lógicos se dividen en dos grandes categorías: combinacionales y secuenciales.

3.1.1. Circuitos Combinacionales

Hussain et al. (2020), sostienen que los circuitos combinacionales son aquellos en los que la salida solo dependía de las entradas actuales, que no poseían memoria interna, que usaban las operaciones lógicas inmediatamente y aquellas deterministas [12].

Las compuertas lógicas básicas AND, OR, NOT, XOR, NAND y NOR son sistemas constitutivos de componentes que se pueden alcanzar, como son los propios sumadores, codificadores, decodificadores y multiplexores. Como mencionan Jafari et al. (2025), las compuertas NAND y NOR son universales y son los componentes esenciales de las tecnologías de lógica reversible para la computación cuántica y la nanoelectrónica [13] Las funciones de las compuertas AND, OR, NOT y XOR se resumen en la *Tabla 1*.

Tabla 2. Compuertas lógicas básicas y sus operaciones

Compuerta	Símbolo	Operación lógica	Tabla de verdad (A, B)
AND	A ∧ B	A y B	0 0 → 0 0 1 → 0 1 0 → 0 1 1 → 1
OR	A ∨ B	A o B	0 0 → 0 0 1 → 1 1 0 → 1 1 1 → 1
NOT	¬A	No A	0 → 1 1 → 0
XOR	A ⊕ B	A distinto B	0 0 → 0 0 1 → 1 1 0 → 1 1 1 → 0

La *Figura 1* muestra los símbolos gráficos utilizados para representar estas compuertas.

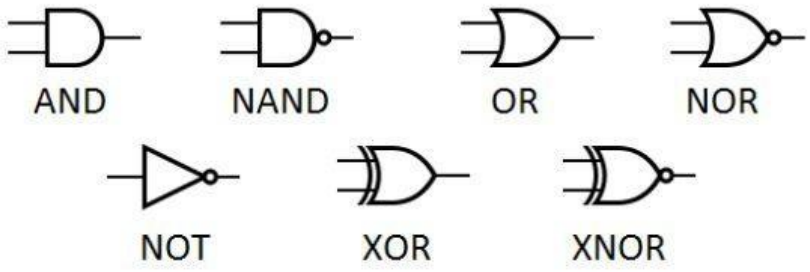


Figura 7. Símbolos de compuertas lógicas

En cuanto a Guitarra et al. (2023), muestran que las tecnologías de memristores pueden permitir la implementación de circuitos para la construcción de sumadores digitales que presentan elevada ratio de fiabilidad, bajo consumo energético y el bajo coste de la unidad aritmética y lógica [14]

3.1.2. Circuitos Secuenciales

Alharbi et al. (2023) discuten que los circuitos secuenciales tienen dentro de sí la memoria, es decir, su salida depende no sólo de las entradas que estén en estado presente, sino también de las condiciones que haya anterior y que por lo tanto son imprescindibles en control, el cronometrado, el almacenamiento y la sincronización [15].

El conjunto de los principales bloques secuenciales es biestable (flip-flops), contadores, registros y máquinas de estado finito (FSM). Repe et al. (2023), se refieren a los flip-flops de tipo SR, D, T y JK como el fundamento de estos circuitos [16]. Los diferentes tipos de flip-flops se describen en la *Tabla 2*, junto con sus entradas, salidas y funciones.

Tabla 3. Tipos de flip-flops y sus funciones

Tipo de Flip-Flop	Entradas	Salidas	Función principal
SR	S, R	Q, Q'	Set/Reset
D	D, CLK	Q, Q'	Almacena dato en flanco
T	T, CLK	Q, Q'	Conmuta si T=1
JK	J, K, CLK	Q, Q'	Versión robusta de SR

La *Figura 2* muestra visualmente la estructura y funcionamiento básico de estos flip-flops.

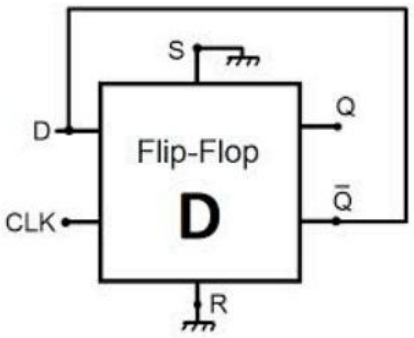


Figura 8. Representación de flip-flop tipo SR, D, T y JK

Mohammadi et al. (2025) destacan que las tecnologías QCA permiten mediante el diseño de flip-flops y contadores un consumo energético y una densidad muy buenas, ideales para IoT y la computación embebida [17]

En la *Figura 3* se ilustra un ejemplo de diseño de circuitos secuenciales aplicando tecnología QCA.

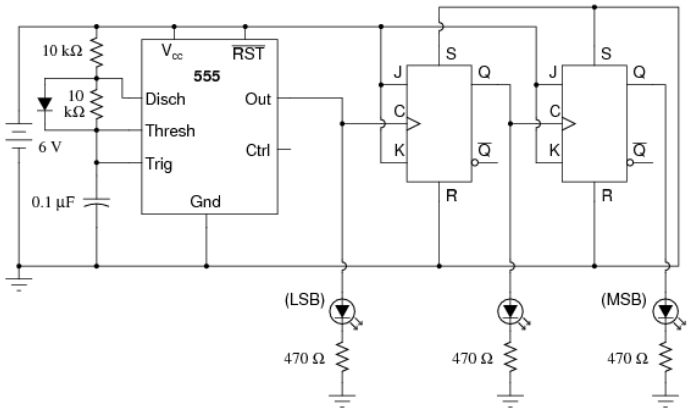


Figura 9. Diseño de circuitos secuenciales con tecnología QCA

3.2. Comparación y Diseño de Sistemas Combinacionales Básicos

3.2.1. Sumadores

Lo que los autores Hussain et al. (2020) ponen de manifiesto es que los sumadores son necesarios para la ejecución de operaciones aritméticas binarias mediante un medio sumador

para la suma simple o un sumador completo para la suma con acarreo [12]. La *Tabla 3* resume las entradas, salidas y ecuaciones lógicas utilizadas en los sumadores más comunes.

Tabla 4. Tipos de sumadores y sus ecuaciones

Tipo de sumador	Entradas	Salidas	Ecuaciones
Half-Adder	A, B	S, Cout	$S = A \oplus B$, $Cout = A \wedge B$
Full-Adder	A, B, Cin	S, Cout	$S = A \oplus B \oplus Cin$, $Cout = (A \wedge B) \vee (Cin \wedge (A \oplus B))$

Un circuito sumador es un circuito que permite la operación aritmética de la suma entre dos bits. Como se observa en la figura 4 existen dos tipos fundamentales de los mismos:

- Half Adder (Sumador Medio): Se encarga de sumar dos bits (A y B), produce una suma (S) y la salida de acarreo (C) [12].
- Full Adder (Sumador Completo): Suma 3 bits (A, B y Cin) considerando el acarreo de entrada y es básico para operaciones en multibit [12]

La *Figura 4* presenta los diagramas de los circuitos de un medio sumador y un sumador completo.

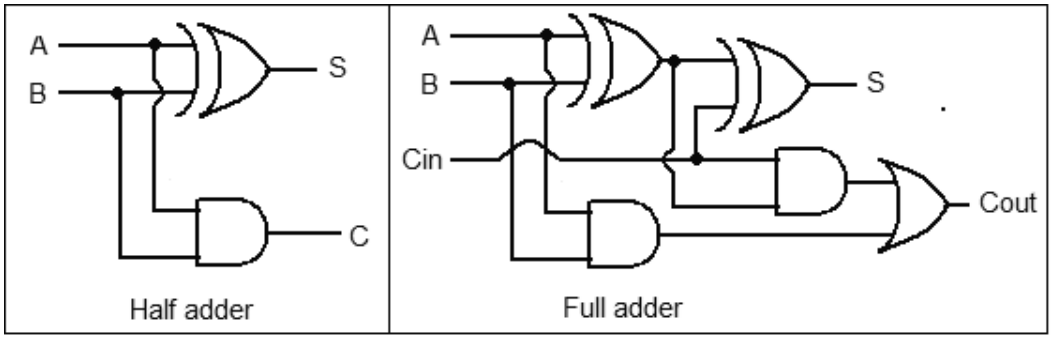


Figura 10. Circuitos de sumador medio y sumador completo

Hamayun et al. (2023) evidencian que la técnica Gate Diffusion Input (GDI) reduce la complejidad, retardo y consumo energético en sumadores [12].

3.2.2. Restadores

Stefanidis et al. (2024) afirman que los restadores se pueden implementar utilizando sumadores que ya están adaptados por complemento a dos, técnica la cual optimiza reutilización de hardware y recursos [18]. En la *Tabla 4* se comparan dos estrategias de diseño para circuitos restadores: clásico y por complemento a dos.

Tabla 5. Comparación entre restador clásico y por complemento a dos

Característica	Restador Clásico	Restador por Complemento a Dos
Estrategia de diseño	Implementación directa con compuertas	Uso de sumador + inversión de bits + acarreo inicial
Componentes principales	XOR, AND, NOT, compuertas para resta directa	Sumador binario, inversor, generador de acarreo inicial
Reutilización de hardware	Limitada	Alta (reutiliza el sumador existente)
Complejidad de implementación	Media	Baja a media (dependiendo del sumador base)
Eficiencia en hardware	Menor eficiencia	Mayor eficiencia, menos elementos redundantes

Velocidad	Variable según diseño	Alta en arquitecturas optimizadas
Uso en ALU modernas	Menos común	Muy común (especialmente en microprocesadores y DSP)

Los restadores efectúan la operación inversa a la de la suma. Al igual que los sumadores se consideran las versiones más simples:

- Half Subtractor (Restador medio): Resta dos bits y devuelve la diferencia y el bit de préstamo [14]
- Full Subtractor (Restador Completo): Realiza la operación de una resta considerando un bit de préstamo anterior, análogo al acarreo en el caso de los sumadores [14]

La *Figura 5* ilustra las configuraciones básicas de medio restador y restador completo.

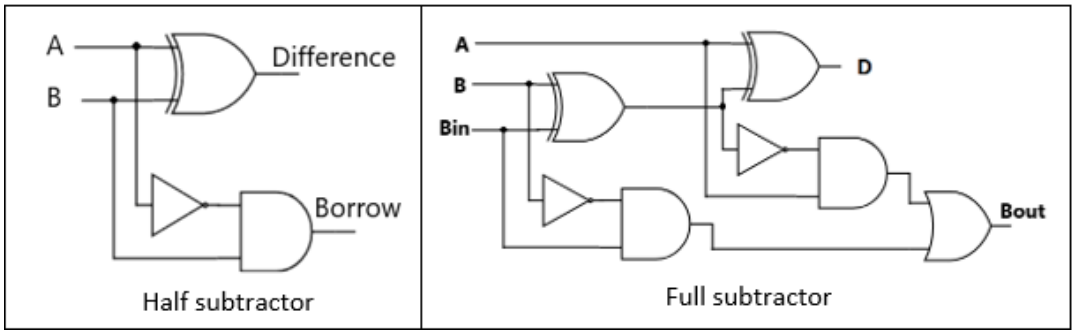


Figure 11. Circuitos de medio restador y restador completo

3.2.3. Multiplexores

Baker et al. (2021) indican que los multiplexores permiten que una entrada sea seleccionada entre varias entradas en función de las señales de control, siendo importantes para direccionar la información. Los multiplexores estocásticos permiten que la precisión y eficiencia energética de los circuitos lógicos en FPGA sean mejorados [19]. La *Tabla 5* muestra las entradas, señales de control y salidas del multiplexor 2:1.

Tabla 6. Configuración de un multiplexor 2:1

MUX	Entradas	Señales de control	Salida
2:1	A, B	S	Y

En la *Figura 7* se representa gráficamente el funcionamiento de un MUX 2:1.

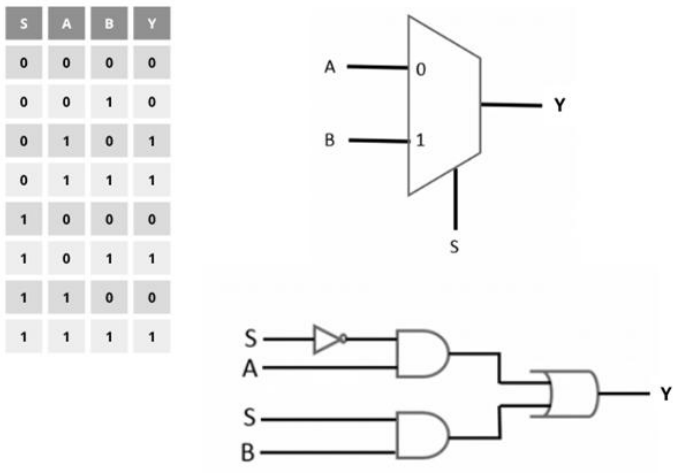


Figura 12. Multiplexor 2:1

3.3. Automatización del Diseño Lógico

Lo evidencian Roy et al. (2021) quien como se aduce "el diseño digital también puede ser automatizado aprovechando algoritmos de autoaprendizaje para optimizar parámetros como tiempo, área, consumo, latencia" [11]

Tal como describen Lin et al. (2021) en cuanto a NovelRewrite "en vez de rediseñar completamente una red lógica existente, el proceso ahora puede ser optimizado facilitando iteraciones entre mejora total de diseño y mejora de diseño directo logrando mejorar los parámetros de diseño" [10]

Ardakani et al. (2025) exponen "Recurrent CircuitSAT Sampling (RCSS) extensible a una arquitectura de computación paralela basada en GPUs, el método asegura que la verificación funcional de circuitos complejos sea un proceso más rápido en cuanto a conjuntos" [20]

Zuo et al. (2025) junto con Xiao et al. (2024) proponen que RL-MUL 2.0 y PrefixLLM emplean modelos de lenguaje, así como aprendizaje automático, para poder sintetizar estructuras aritméticas, logrando alta eficiencia y menor tiempo de desarrollo [21], [22].

Lai et al. (2024) proponen ArithTreeRL para lograr construir árboles aritméticos adaptativos que optimizan la profundidad y la capacidad paralela, que le aportan de este modo beneficios al procesamiento digital de señales [23].

4. MICROPROCESADOR Y MICROCONTROLADORES

4.1. Microprocesador

De acuerdo con el estudio de Abdullayevich (2020), el microprocesador es el componente principal de la computadora. Tiene un papel fundamental, ya que es capaz de ejecutar todas las operaciones aritméticas y lógicas indicadas por un programa, además de controlar el proceso de cálculo y coordinar la ejecución de los demás dispositivos del sistema. De este modo, viene a ser el cerebro que envía señales necesarias para controlar las distintas partes de la computadora y se asemeja a la función del cerebro humano que controla el cuerpo. Este componente es capaz de realizar operaciones a gran velocidad [24].

Según el estudio de Khan, Pasha, y Masud (2021), el desarrollo de los microprocesadores se debe a la creación de los transistores semiconductores por Bell Labs en 1947, y la invención de los chips de circuitos integrados (IC) por Robert Noyce en 1961. Busicom, una empresa de calculadoras, en 1969 contactó a Intel para encargar 12 chips. Intel como respuesta, propuso

cuatro diseños, en el cual uno de ellos permitía programarse de diferentes maneras para así satisfacer los requisitos del cliente [25].

4.1.1. Arquitectura básica

La arquitectura de un microprocesador, como el µPD propuesto en el artículo "Building a Microprocessor Architecture at Computer Engineering Undergraduate Courses" (2020), se basa en un diseño monociclo RISC (Reduced Instruction Set Computer) con un bus de datos de 16 bits. Básicamente, un microprocesador se conecta a una memoria de programa externa (ROM) en la que se almacena el código máquina y tiene buses de entrada/salida para comunicarse con las demás unidades (dispositivos) externos y atender interrupciones. Se divide internamente en ruta de datos y ruta de control. La ruta de control se encarga de la lectura de las instrucciones y de gestionar el flujo de datos. Contiene un bloque de control y una memoria LIFO para las subrutinas e interrupciones. La ruta de datos tiene elementos esenciales como un bloque de registros de propósito general, una Unidad Aritmética Lógica (ALU) para operaciones matemáticas y lógicas, y registros individuales para transferir datos de entrada y salida. Se distingue la gestión entre la memoria de programa (ROM) y la de datos (RAM), siendo esta última utilizada para almacenar hasta 1K palabras de 16 bits, es decir, 2048 bytes (2K en base binaria) de capacidad de almacenamiento [26].

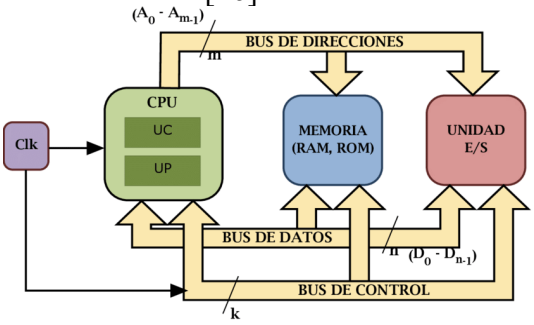


Figura 13. Arquitectura del microprocesador

4.2. Microcontrolador

En el estudio de Samiullah, Irfan y Rafique (2023) se destaca que los microcontroladores son un elemento fundamental en los sistemas integrados, los cuales constituyen el núcleo de diversas aplicaciones, permitiendo su desarrollo en múltiples ámbitos. El motivo por el cual son tan requeridos es por su tamaño reducido y su bajo consumo de energía. En la actualidad, son indispensables pues, se pueden encontrar tanto en los circuitos que utilizamos cotidianamente en nuestra vivienda como en sistemas de control automatizado industrial [27].

4.2.1. Arquitectura básica

El artículo "Sensors and Microcontroller Unit (MCU)-Based Data Acquisition System for Smart Greenhouses" (2020) ofrece una clara descripción de la arquitectura del microcontrolador, ya que muestra cómo están distribuidos sus componentes dentro del chip. Para este propósito, hace uso del ESP32. Su arquitectura básica se centra en una CPU (Unidad Central de Proceso) de doble núcleo de 32 bits, que trabaja síncronamente a partir de una señal de reloj de hasta 240 MHz. Esta CPU está específicamente interconectada con bloques de memorias integradas: 520 KB de SRAM para el almacenamiento volátil de datos en tiempo de ejecución, y 4 MB de Flash para almacenar el programa y configuraciones en modo no volátil de memoria. La principal característica de la arquitectura de los microcontroladores es la integración de diferentes componentes en un solo chip, donde el microcontrolador incluye un conjunto amplio de Periféricos de Entrada/Salida (GPIOs, ADCs, DACs), módulos de comunicación (UART, SPI, I2C), entre otros, para que puedan interactuar con la CPU y con la memoria a través de buses internos [28].

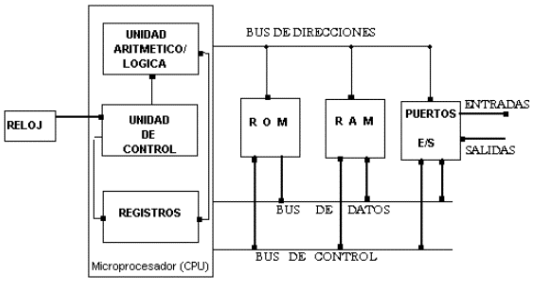
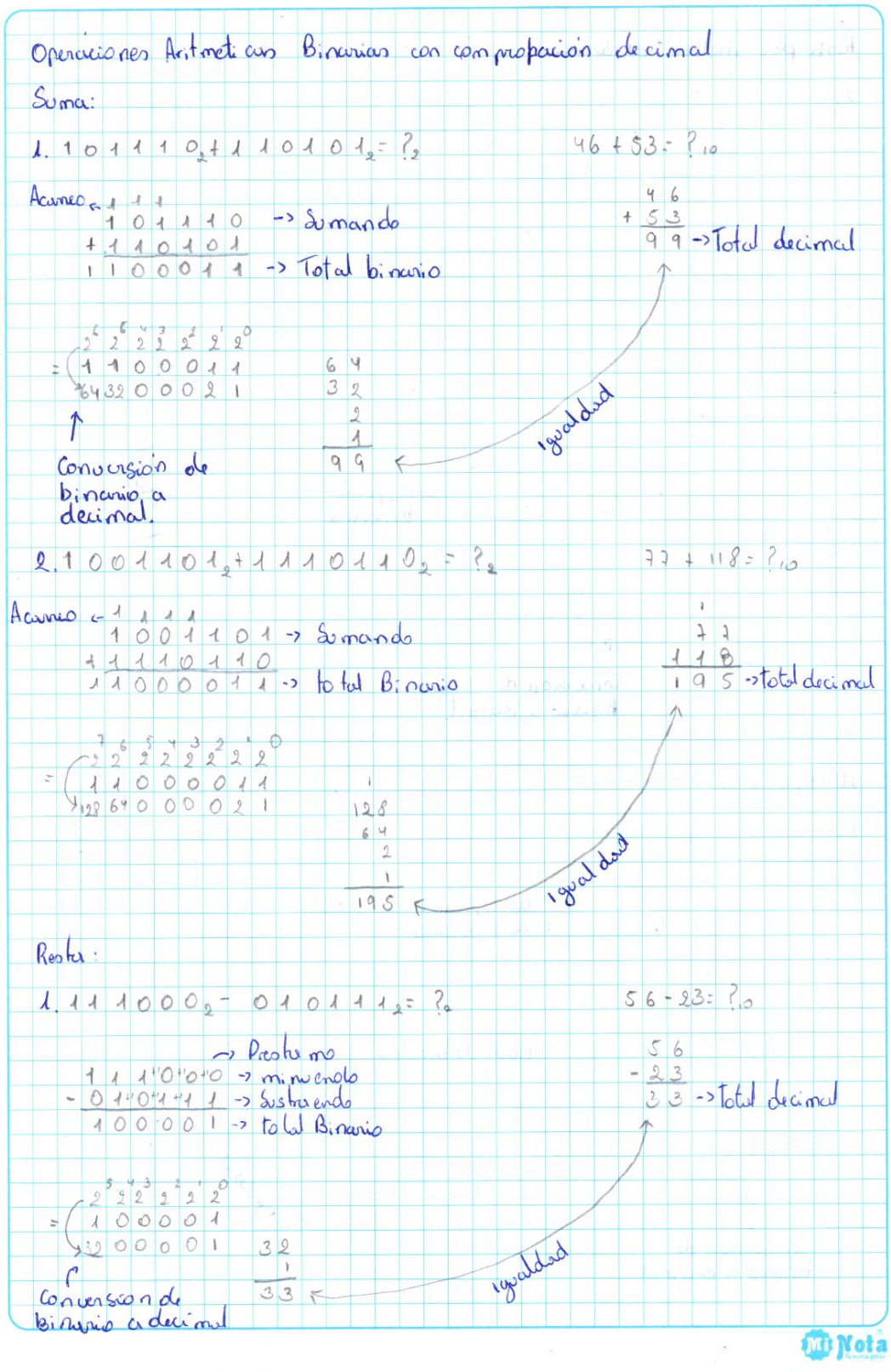


Figura 14. Arquitectura del microcontrolador

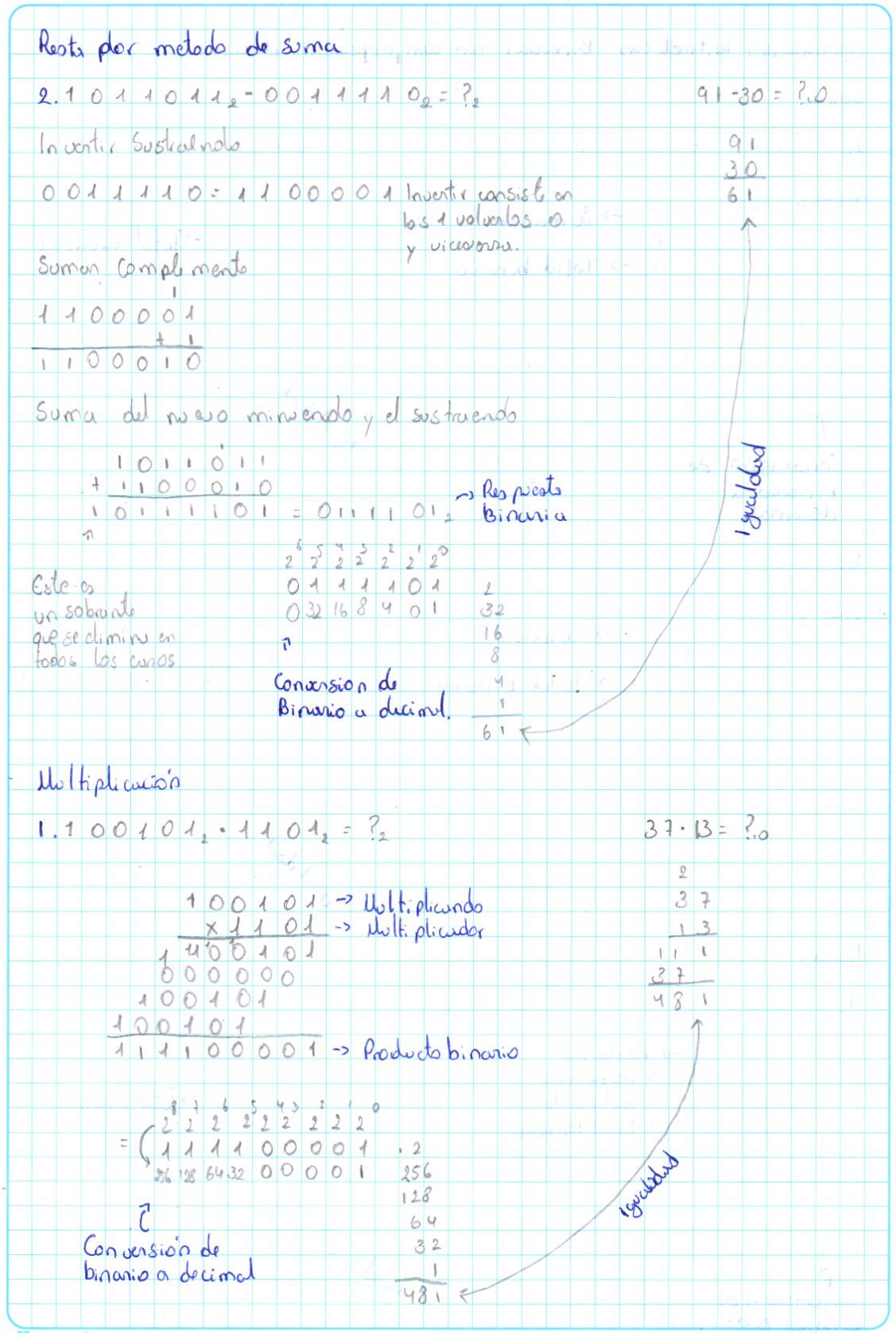
5. EJERCICIOS

5.1. Operaciones de Aritmética Binaria y Decimal:

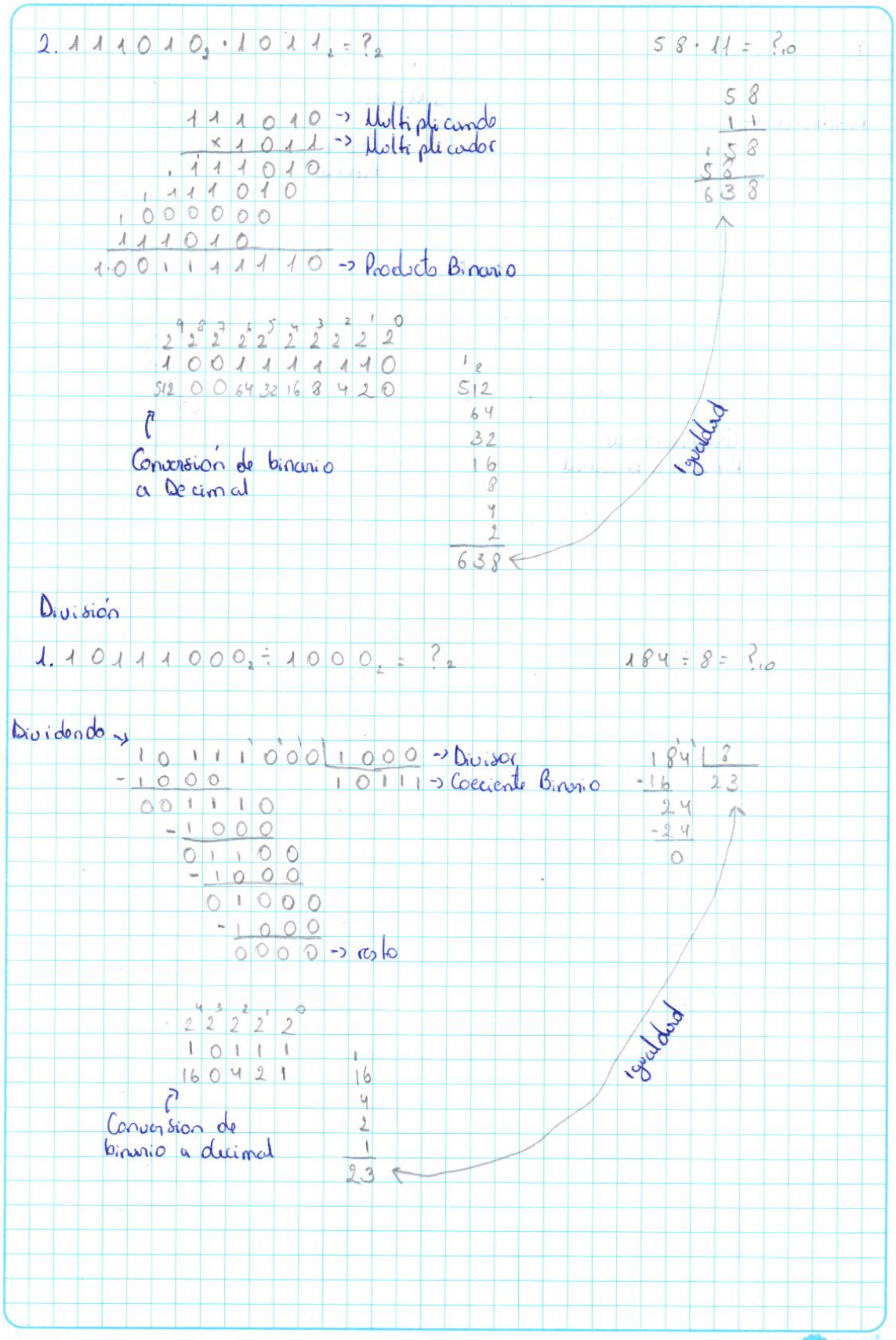
Suma



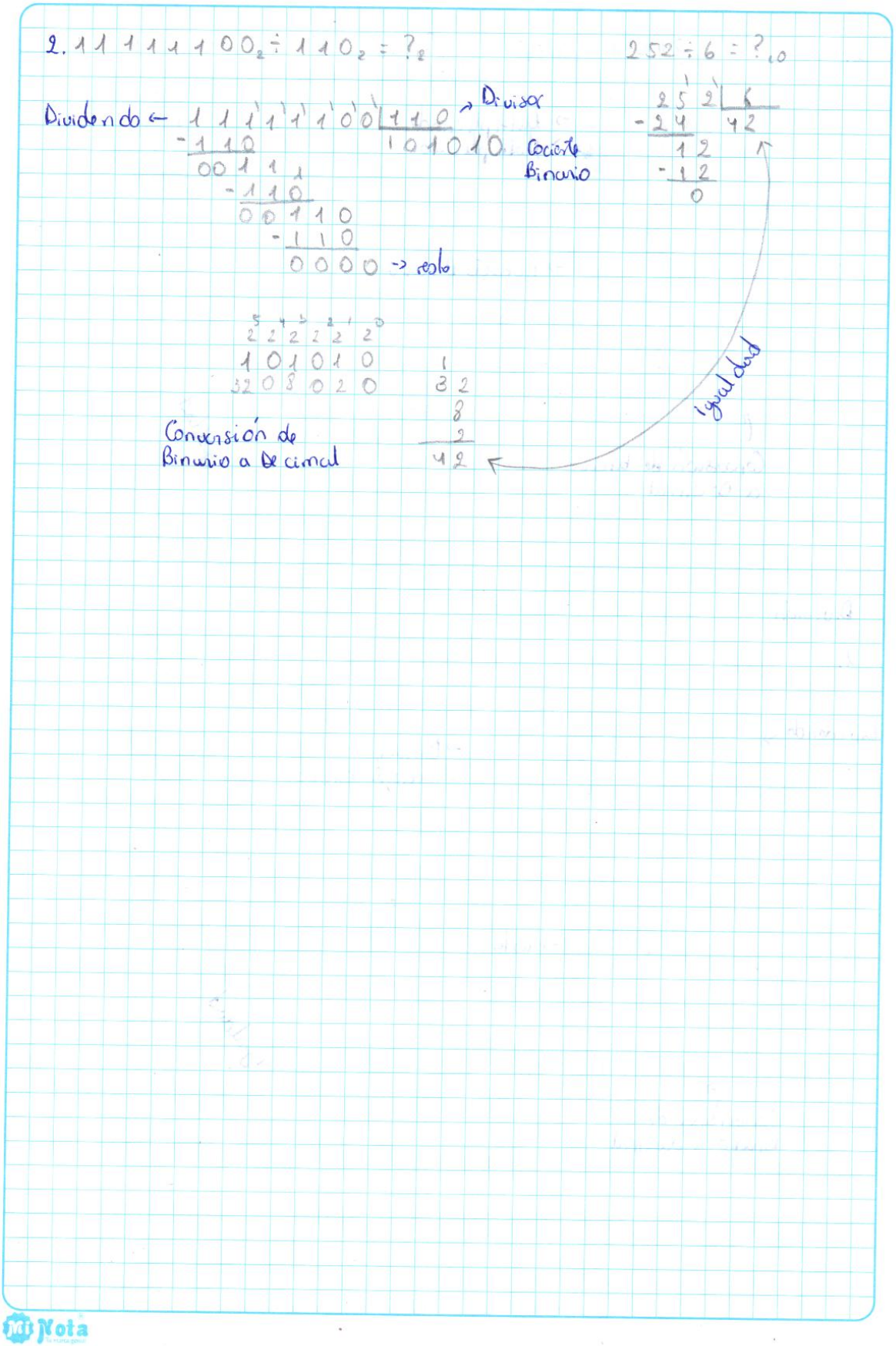
Resta


Resta por metodo de suma
2.1011011₂ - 0011110₂ = ?₂
91 - 30 = ?.0
Invertir Sustraendo
0011110 = 1100001
Invertir consiste en los 1 volverlos 0 y viceversa.
Sumar Complemento
1100001
+ 1
1100010
Suma del nuevo minuendo y el sustraendo
1011011
+ 1100010
10111101 = 01111012
-> Respuesta Binaria
Este es un sobrante que se elimina en todos los casos
Conversion de Binario a decimal
Igualdad
Multiplicación
1.1001012 · 11012 = ?2
37 · 13 = ?.0
1001012 -> Multiplicando
x1101 -> Multiplicador
1001012
0000000
1001012
1001012
11110000012 -> Producto binario
Conversion de binario a decimal
Igualdad

Multiplicación



División


2. 11111100₂ ÷ 110₂ = ?₂
Dividendo
Divisor
Cociente Binario
-110
001111
-110
00110
-110
0000 -> residuo
252 ÷ 6 = ?₁₀
25 2 | 6
-24 42
12
-12
0
Igualdad
Conversión de Binario a Decimal
32
8
2
42

5.2. Ejercicios de Álgebra Booleana

Actividad

- Realizar una simplificaciones de expresión usando mapas de Karnough.
- Expresion de 4 variables (A, B, C, D)

$F_1 = \bar{B}C\bar{D} + A\bar{C}D + B\bar{D} + \bar{A}B\bar{C}$

1: Grupo de cada expresión

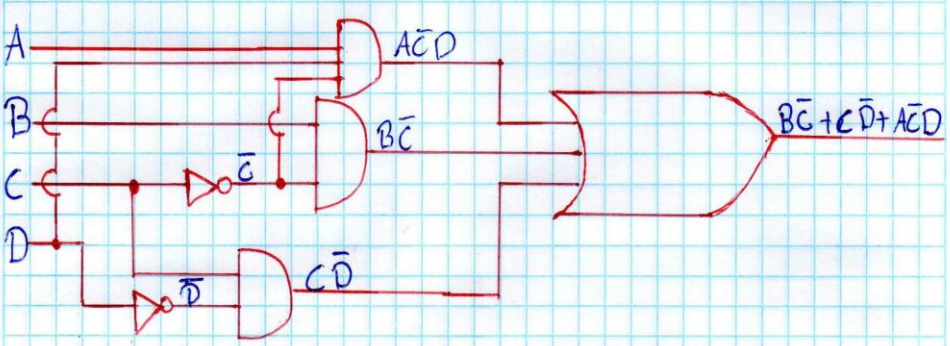
- $\bar{B}C\bar{D} = X010$
  $= 1010 - 0010$
- $A\bar{C}D = 1X01$
  $= 1001 - 1101$
- $B\bar{D} = X1X0$
  $- 0100 - 1100$
  $- 0110 - 1110$
- $\bar{A}B\bar{C} = 010X$
  $= 0100 - 0101$

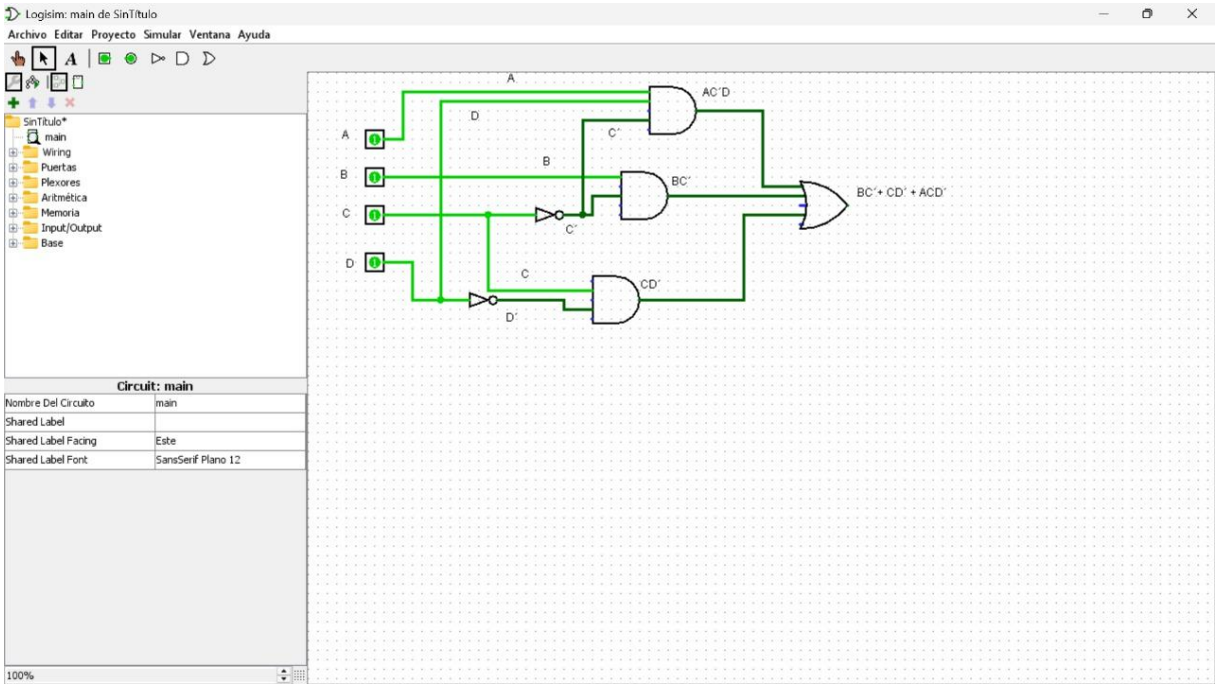
2: Mapa de Karnough

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	1
$\bar{A}B$	1	1	0	1
AB	1	1	0	1
$A\bar{B}$	0	1	0	1

$F_1 = B\bar{C} + C\bar{D} + A\bar{C}D$

3. Circuito




Logisim: main de SinTítulo
Archivo Editar Proyecto Simular Ventana Ayuda
SinTítulo*
main
Wiring
Puertas
Plexores
Aritmética
Memoria
Input/Output
Base
Circuit: main
Nombre Del Circuito
main
Shared Label
Shared Label Facing
Este
Shared Label Font
SansSerif Plano 12
100%
A
B
C
D
AC´D
BC´
CD´
BC´+ CD´ + ACD´

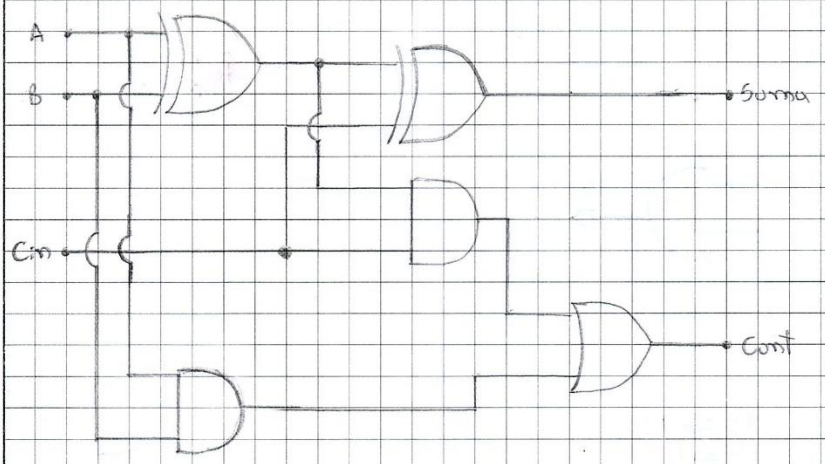
5.3. Diseño de Circuitos Combinacionales

Construcción de un sumador binario completo

Diseño de circuitos combinacionales

- Construcción de un sumador binario completo

A	B	Cin	Cout	Suma
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

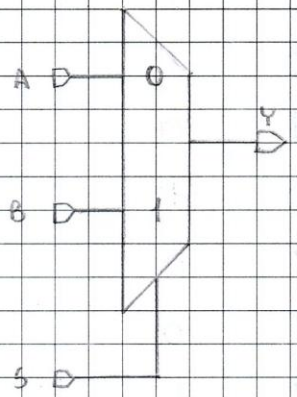


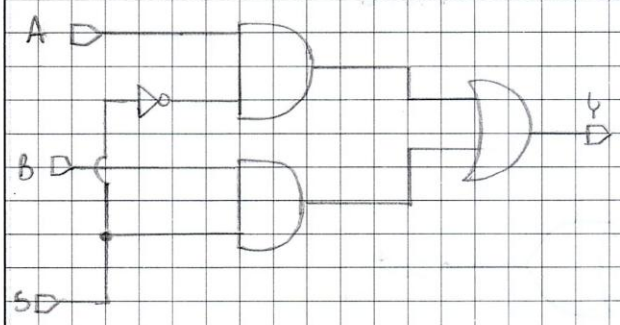
Implementar un multiplexor simple para controlar el flujo de datos

- Implementar un multiplexor simple para controlar el flujo de datos

Multiplexor de 2 entradas

S	A	B	S_1	S_0	Y
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1





Si S = 0:
Y ← A

Si S = 1:
Y ← B

5.4. Estudio de Microprocesadores y Microcontroladores

5.4.1. Ciclo de instrucciones del microprocesador

El artículo titulado "Design and Implementation of a 256-Bit RISC-V-Based Dynamically Scheduled Very Long Instruction Word on FPGA" expresa el modelo de ciclo de instrucción para su microprocesador VLIW (Very Long Instruction Word) basado en RISC-V, que consiste en un conjunto de seis etapas enlazadas entre sí. El ciclo empieza en la etapa Fetch (búsqueda) de una instrucción, para luego introducir una fase clave del Instruction Scheduler (planificador de instrucciones) que, a diferencia de los diseños VLIW estándar, es la responsable de empaquetar dinámicamente instrucciones independientes, generando un bloque de instrucciones que se corresponde con 256 bits, por lo que el uso de un compilador VLIW se puede considerar innecesario. Posteriormente, la instrucción empaquetada es decodificada a través de la etapa Decode (decodificación) para que el procesador pueda interpretarla. La Execute (ejecución) es el momento en que la multiplicidad de operaciones que alberga la instrucción VLIW tiene lugar de forma paralela. Tras ello, en caso de que la instrucción lo requiera, da acceso a la Data Memory (memoria de datos) en la etapa correspondiente. Para terminar, en la etapa Writeback (escritura de resultados), los resultados de las operaciones se almacenan de nuevo en los registros o la memoria. Se completa el ciclo y el procesador queda preparado para la siguiente instrucción [29].

5.4.2. Velocidad de procesamiento

De acuerdo con Lei Wang (2020) los datos procesados por las computadoras se consideran muy importantes. Los factores principales que tienen relación con ello son la CPU, cuya frecuencia principal tiene una relación directa con la rapidez con que se efectúan las operaciones. La memoria, que permite la mejora de efectividad mediante el caché; y el disco duro, el cual también se considera que también juega un papel importante en la velocidad de lectura [30].

5.4.3. Analizar el ciclo de instrucciones y la velocidad del microprocesador

Un microprocesador tiene una frecuencia de 2.5 GHz y se desea analizar cuánto tiempo tarda en ejecutar el siguiente bloque de instrucciones:

Code (Instruction Set)

```
; Programa que suma dos números y muestra el resultado
MOV A, 5      ; Carga el valor 5 en el registro A
MOV B, 3      ; Carga el valor 3 en el registro B
ADD A, B      ; Suma A + B (resultado en A)
MOV [0], A    ; Guarda el resultado en la posición de memoria 0
HLT           ; Detiene la ejecución.
```

Figura 13. Código de Assembler

5.4.4. Contar las instrucciones y ciclos

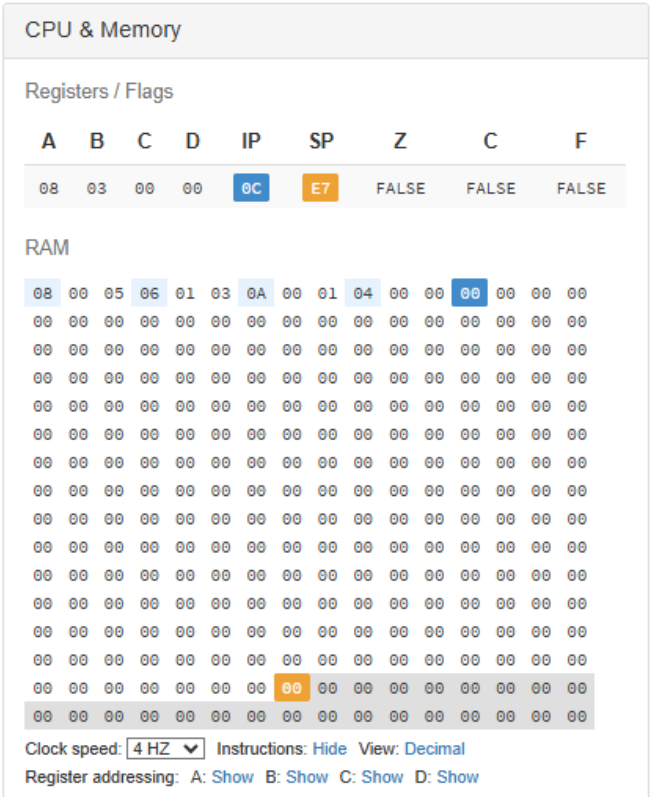


Figure 14. Registros

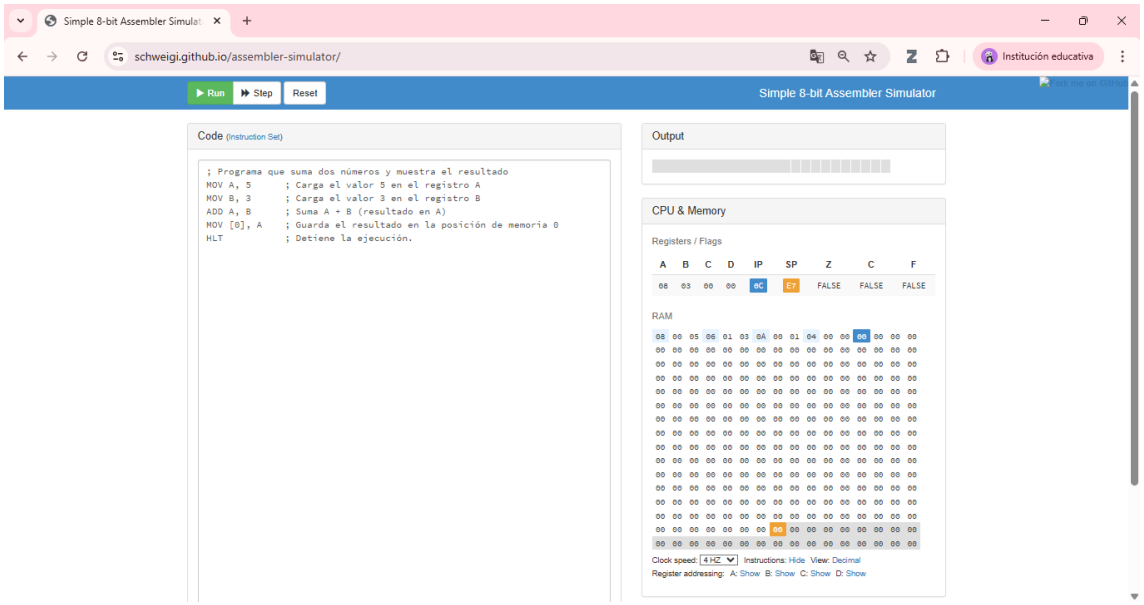


Figure 15. Assembler

Números de instrucciones: 4

Total de ciclos: 3 + 4 + 3 + 3 = 14

- **Calcular los Ciclos por instrucción (CPI)**

$$CPI = \frac{Total\ de\ ciclos}{Número\ de\ instrucciones} = \frac{13}{4} = 3.25$$

- **Calcular tiempo de ciclo**

$$Tiempo\ de\ ciclo = \frac{1}{2.5\ GHz} = \frac{13}{2.5 \times 10^9} = 0.4\ ns$$

- **Calcular el tiempo de ejecución**

$$Tiempo\ de\ ejecución = CPI \times número\ de\ instrucciones \times tiempo\ de\ ciclo$$
$$= 3.25 \times 4 \times 0.4\ ns = 5.2\ ns$$

REFERENCIAS

[1] A. Falconi Asanza and F. M. Hernandez Crespo, *Matematica en espiral*. Editorial Universo Sur, 2020. [Online]. Available: https://elibro.net/es/lc/uteq/titulos/131900

[2] M. D. P. Alegre Ramos, "Sistemas informáticos - ALEGRE RAMOS, MARIA DEL PILAR - Google Libros." Accessed: Jul. 25, 2025. [Online]. Available: https://books.google.com.ec/books?hl=es&lr=&id=kR3JEAAAQBAJ&oi=fnd&pg=PR5&dq=operaciones+aritmeticas+en+sistema+binario&ots=zC-oEu4WFi&sig=Da1lp3Sovxp1lTxp145HlOrnDqs&redir_esc=y#v=onepage&q&f=false

[3] P. G. Recabarren, *Introduccion a la electronica digital: teoria, circuitos y ejercicios de aplicacion*. Jorge Sarmiento Editor - Universitas, 2020. [Online]. Available: https://elibro.net/es/lc/uteq/titulos/172319

[4] A. Chandra Jha, "NJ: NUTA Positional Number System," 2020.

[5] A. Elahi, *Computer Systems: Digital Design, Fundamentals of Computer Architecture and ARM Assembly Language: Second Edition*. Springer International Publishing, 2022. doi: 10.1007/978-3-030-93449-1.

[6] E. M. Jiménez-Hernández, H. Oktaba, F. Díaz-Barriga, and M. Piattini, "Using web-based gamified software to learn Boolean algebra simplification in a blended learning setting," *Computer Applications in Engineering Education*, vol. 28, no. 6, pp. 1591–1611, Nov. 2020, doi: 10.1002/cae.22335.

[7] J. Salido, "Lógica digital y Tecnología de Computadores - Un enfoque práctico mediante simulación con Logisim."

[8] Carpinelli, "This page intentionally left blank," 2023, doi: 10.60826/kr55-0k56.

[9] B. J. Lameres, "Introduction to Logic Circuits & Logic Design with VHDL."

[10] S. Lin, J. Liu, T. Liu, M. D. F. Wong, and E. F. Y. Young, "NovelRewrite," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, New York, NY, USA: ACM, Jul. 2022, pp. 427–432. doi: 10.1145/3489517.3530462.

[11] R. Roy *et al.*, "PrefixRL: Optimization of Parallel Prefix Circuits using Deep Reinforcement Learning," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, IEEE, Dec. 2021, pp. 853–858. doi: 10.1109/DAC18074.2021.9586094.

[12] I. Hussain and S. Chaudhury, "Fast and High-Performing 1-Bit Full Adder Circuit Based on Input Switching Activity Patterns and Gate Diffusion Input Technique," *Circuits Syst Signal Process*, vol. 40, no. 4, pp. 1762–1787, Apr. 2021, doi: 10.1007/s00034-020-01550-3.

[13] M. Jafari, S. Sayedsalehi, R. Faghih Mirzaee, and R. Farazkish, "Design of high-performance quaternary half adder, full adder, and multiplier," *Analog Integr Circuits Signal Process*, vol. 122, no. 2, p. 25, Feb. 2025, doi: 10.1007/s10470-025-02317-z.

[14] S. Guitarra, R. Taco, M. Gavilánez, J. Yépez, and U. Espinoza, "Assessment of a universal logic gate and a full adder circuit based on CMOS-memristor technology," *Solid State Electron*, vol. 207, p. 108704, Sep. 2023, doi: 10.1016/j.sse.2023.108704.

[15] M. Alharbi, G. Edwards, and R. Stocker, "Novel ultra-energy-efficient reversible designs of sequential logic quantum-dot cellular automata flip-flop circuits," *J Supercomput*, vol. 79, no. 10, pp. 11530–11557, Jul. 2023, doi: 10.1007/s11227-023-05134-1.
[16] M. Repe and S. Koli, "Flip Flops Design in Quantum Dot Cellular Automata Technology: Towards Digitization," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, no. 6, pp. 151–158, Jul. 2023, doi: 10.17762/ijritcc.v11i6.7302.
[17] J. Mohammadi, M. Zare, and M. Molaei, "Optimized Three Bit Counter Employing T Flip-Flop in Quantum-Dot Cellular Automata Technology," *Journal of Electrical and Electronic Engineering*, vol. 13, no. 1, pp. 40–45, Feb. 2025, doi: 10.11648/j.jeee.20251301.14.
[18] A. Stefanidis and G. Dimitrakopoulos, "Reinforcement-Learning-Based Synthesis of Custom Approximate Parallel Prefix Adders," *Journal of Low Power Electronics and Applications*, vol. 14, no. 4, p. 57, Dec. 2024, doi: 10.3390/jlpea14040057.
[19] T. J. Baker and J. P. Hayes, "CeMux: Maximizing the Accuracy of Stochastic Mux Adders and an Application to Filter Design," Aug. 2021.
[20] A. Ardakani, K. He, and J. Wawrzynek, "Recurrent CircuitSAT Sampling for Sequential Circuits," Mar. 2025.
[21] D. Zuo, J. Zhu, Y. Ouyang, and Y. Ma, "RL-MUL 2.0: Multiplier Design Optimization with Parallel Deep Reinforcement Learning and Space Reduction," *ACM Transact Des Autom Electron Syst*, Jan. 2025, doi: 10.1145/3711850.
[22] W. Xiao, V. S. C. Putrevu, R. V. Hemadri, S. Garg, and R. Karri, "PrefixLLM: LLM-aided Prefix Circuit Design," Dec. 2024.
[23] Y. Lai, J. Liu, D. Z. Pan, and P. Luo, "Scalable and Effective Arithmetic Tree Generation for Adder and Multiplier Designs," May 2024.
[24] H. Z. Abdullayevich, "History, Structure And Types Of Microprocessors," *The American Journal of Interdisciplinary Innovations and Research*, vol. 02, no. 11, pp. 39–46, Nov. 2020, doi: 10.37547/tajiir/Volume02Issue11-08.
[25] F. H. Khan, M. A. Pasha, and S. Masud, "Advancements in Microprocessor Architecture for Ubiquitous AI—An Overview on History, Evolution, and Upcoming Challenges in AI Implementation," *Micromachines (Basel)*, vol. 12, no. 6, p. 665, Jun. 2021, doi: 10.3390/mi12060665.
[26] M. Sartor, T. T. M. S. Soares, and M. D. Berejuck, "Building a microprocessor architecture at Computer Engineering undergraduate courses," *International Journal of Advanced Engineering Research and Science*, vol. 7, no. 7, pp. 36–48, 2020, doi: 10.22161/ijaers.77.5.
[27] M. Samiullah, M. Z. Irfan, and A. Rafique, "Microcontrollers: A Comprehensive Overview and Comparative Analysis of Diverse Types," Sep. 2023, doi: 10.31224/3228.
[28] Z. Wu, K. Qiu, and J. Zhang, "A Smart Microcontroller Architecture for the Internet of Things," *Sensors*, vol. 20, no. 7, p. 1821, Mar. 2020, doi: 10.3390/s20071821.
[29] N. M. Qui, C. H. Lin, and P. Chen, "Design and Implementation of a 256-Bit RISC-V-Based Dynamically Scheduled Very Long Instruction Word on FPGA," *IEEE Access*, vol. 8, pp. 172996–173007, 2020, doi: 10.1109/ACCESS.2020.3024851.
[30] L. Wang, "Analysis of Factors Affecting Computer Data Processing Speed," *J Phys Conf Ser*, vol. 1648, no. 2, p. 022136, Oct. 2020, doi: 10.1088/1742-6596/1648/2/022136.

6. ANEXOS

ENLACE DE GITHUB:
https://github.com/AnderJM/ArqComp-Grupo-D-Unid-3