

# Curso: Algorítmica y fundamentos de programación



## Semana 01 - Grupo 8

### Integrantes:

- Ayzanoa Solano, Joao Carlos
- Huamani Huaman Victor Jesus
- Salazar Espiritu Rodrigo Ahmed

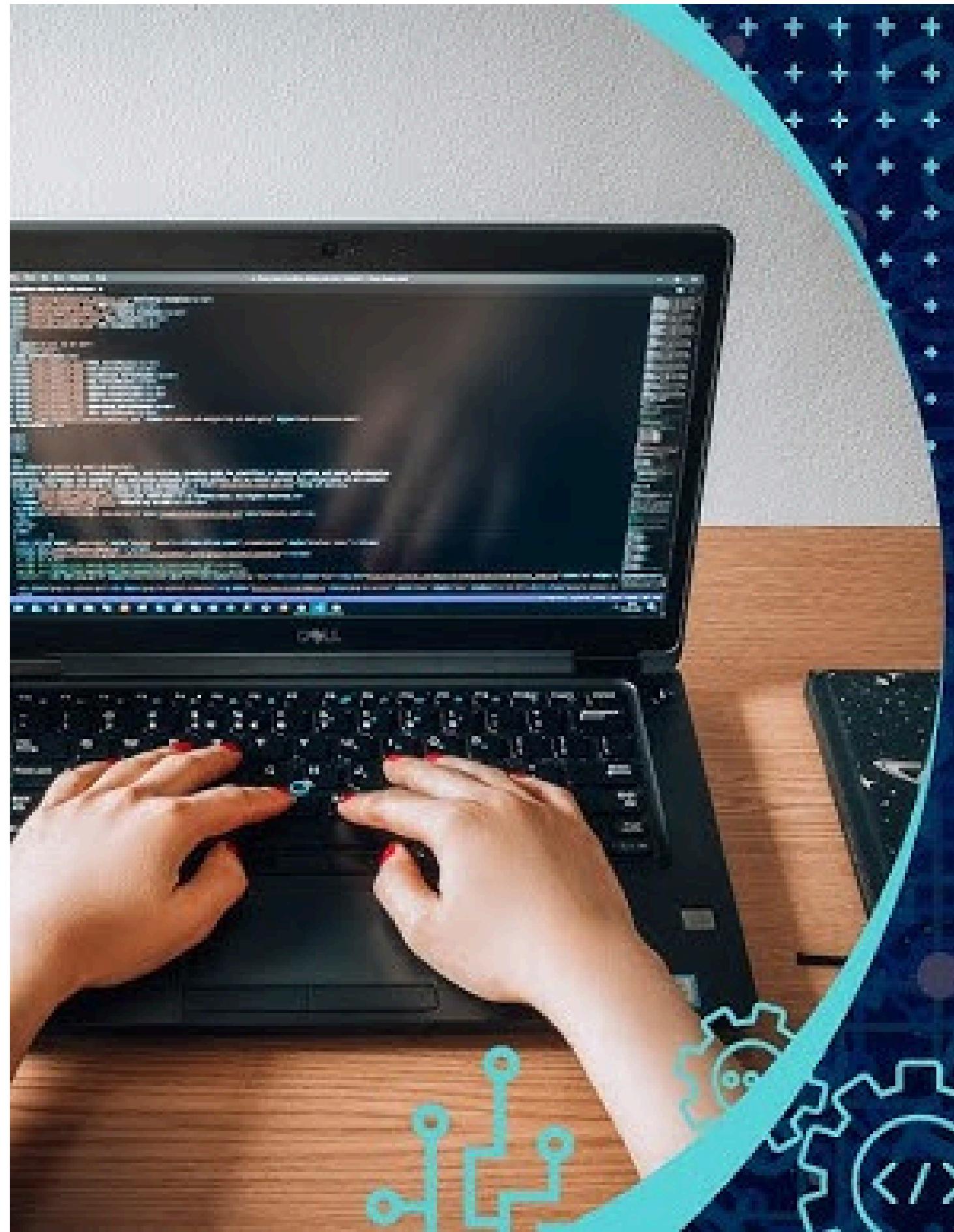
# ÍNDICE DE CONTENIDOS



```
4 # Prevent database truncation if the test fails
5 abort("The Rails environment is running in production mode")
6 require 'spec_helper'
7 require 'rspec/rails'
8
9 require 'capybara/rspec'
10 require 'capybara/rails'
11
12 Capybara.javascript_driver = :webkit
13 Category.delete_all; Category.create(name: "Electronics", description: "Electronics category")
14 Shoulda::Matchers.configure do |config|
15   config.integrate do |with|
16     with.test_framework :rspec
17     with.library :rails
18   end
19 end
20
21 # Add additional requires below this line to access your own code:
22
23 # Requires supporting ruby files with custom matchers and helpers
24 # in spec/support/ and its subdirectories. This directory also
25 # contains supporting files for this command:
26 # - spec/runner/ (for running specs from command line)
27 # - spec/diff (for custom diff output)
28 # run twice. It is recommended that you do not name this file spec/spec.rb
29 # end with _spec.rb. You can configure the :file option in .rspec for
30 # automation or the --require option for manual invocation.
```

INTRODUCCIÓN A LOS FUNDAMENTOS DE LA COMPUTACIÓN	01
ARQUITECTURA DE LA COMPUTADORA	02
REPRESENTACIÓN DE LOS DATOS	03
INTERFAZ HOMBRE COMPUTADOR	04
SISTEMAS OPERATIVOS	05
COMPILADORES Y TRADUCTORES	06

# 01. INTRODUCCIÓN A LOS FUNDAMENTOS DE LA COMPUTACIÓN



# COMPUTACIÓN

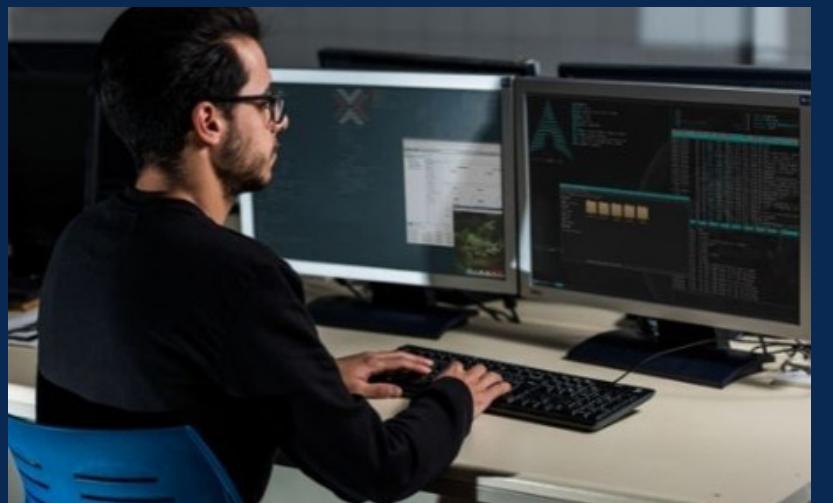
---

## ¿QUÉ ES?





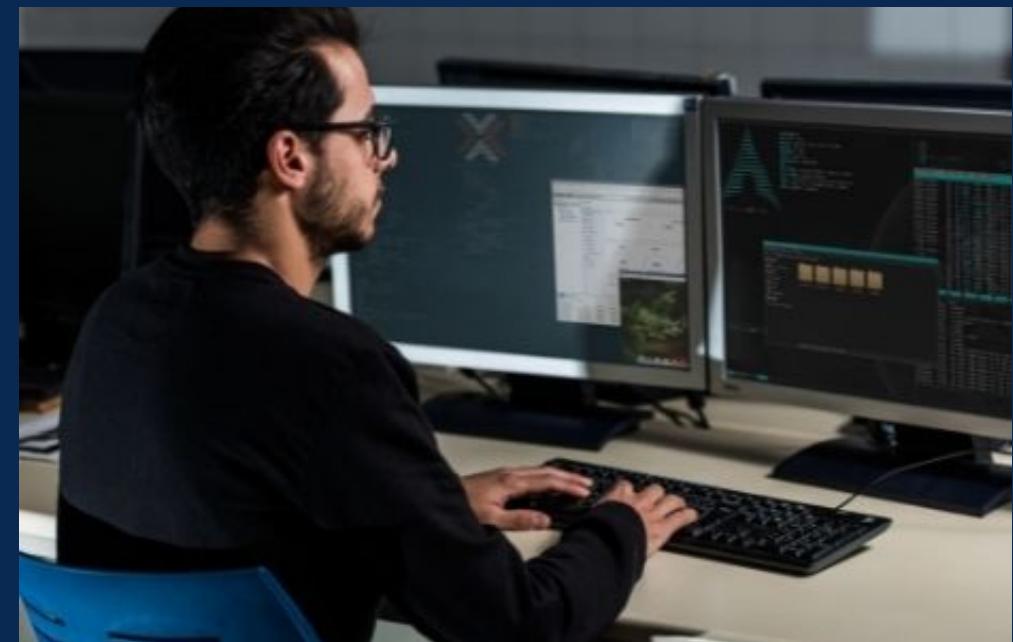
El concepto de computación se refiere al estudio y aplicación de principios, técnicas y tecnologías para el procesamiento automático de información mediante computadoras. Abarca tanto los fundamentos teóricos como los aspectos prácticos del diseño, desarrollo y uso de sistemas informáticos.





# Importancia en la actualidad

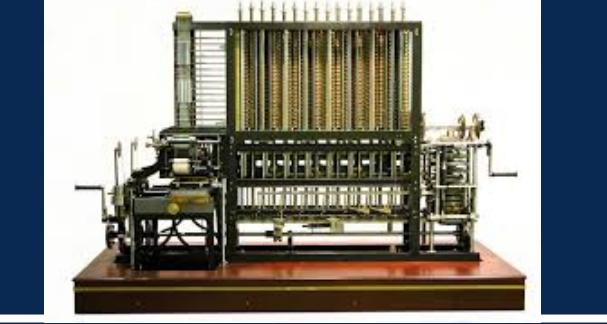
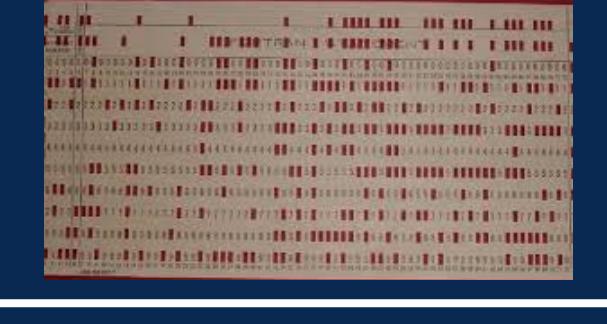
La computación es el pilar de la era digital, permitiendo la automatización, el procesamiento de datos y la innovación en todas las áreas del conocimiento. Su impacto abarca desde la inteligencia artificial y la ciberseguridad hasta la medicina, la educación y la economía, facilitando avances científicos, mejorando la comunicación global y optimizando procesos en la industria. Gracias a la computación, la sociedad evoluciona constantemente, impulsando la creación de nuevas tecnologías que transforman la manera en que trabajamos, aprendemos y nos relacionamos, haciendo posible un futuro más eficiente, interconectado y avanzado.



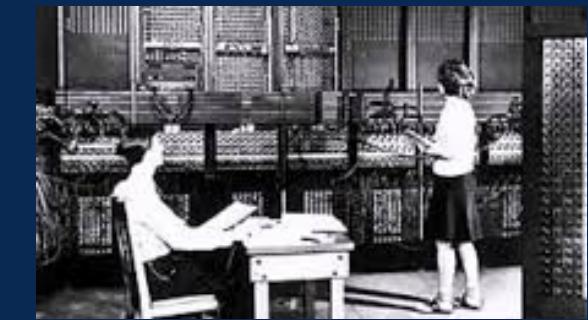
# *El origen de las computadoras y su evolución*



Antes de las computadoras modernas, hubo dispositivos mecánicos que ayudaban en cálculos matemáticos:

Ábaco (siglo V a.C.)	Considerado la primera herramienta de cálculo.	
Máquina de Pascal (1642)	Blaise Pascal diseñó una calculadora mecánica para sumar y restar.	
Máquina Analítica de Babbage (1837)	Charles Babbage diseñó un dispositivo programable con una idea similar a las computadoras actuales, pero nunca se construyó completamente en su época.	
Tarjetas perforadas (siglo XIX)	Usadas en telares y luego en computadoras tempranas para almacenar datos.	

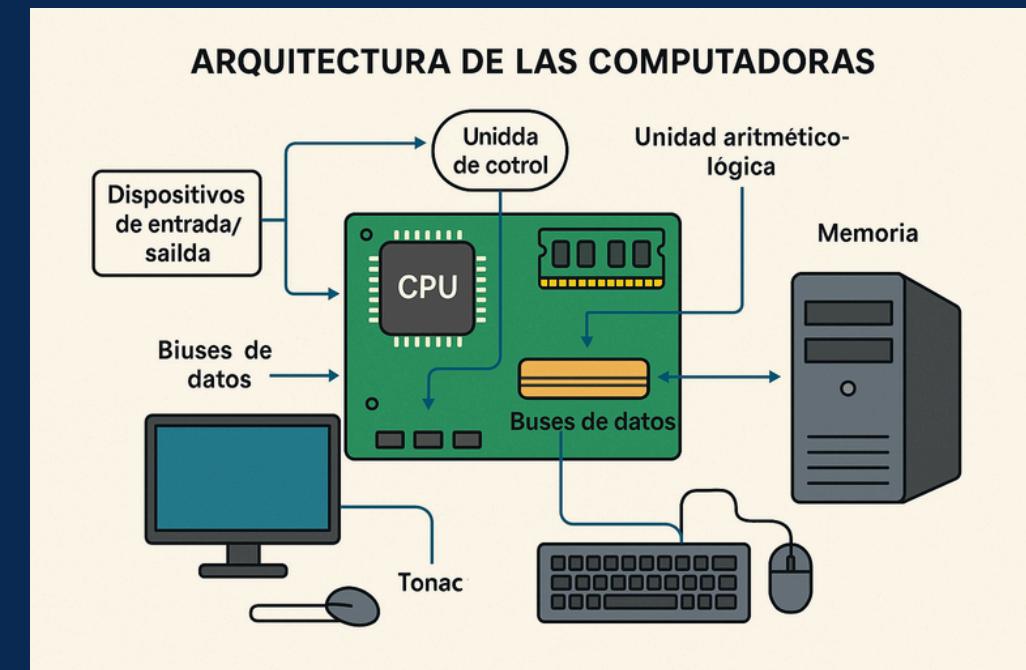
# Generaciones de computadoras

<b>Primera generación (1940-1956) - Computadoras con tubos de vacío</b>	<p>ENIAC (1946): Considerada la primera computadora electrónica de propósito general. Estas máquinas eran grandes, consumían mucha energía y usaban tubos de vacío para procesar datos.</p>	
<b>Segunda generación (1956-1964) - Computadoras con transistores</b>	<p>Reemplazo de los tubos de vacío por <b>transistores</b>, lo que redujo el tamaño y mejoró la eficiencia. Primeros lenguajes de programación como <b>FORTRAN</b> y <b>COBOL</b>.</p>	
<b>Tercera generación (1964-1971) - Circuitos integrados</b>	<p>Uso de <b>circuitos integrados (chips)</b> que permitieron computadoras más pequeñas y rápidas. Popularización del <b>uso comercial</b> de las computadoras.</p>	
<b>Cuarta generación (1971-presente) - Microprocesadores</b>	<p>Aparece el <b>microprocesador</b> (1971, Intel 4004), integrando todos los circuitos en un solo chip. Nacimiento de las <b>computadoras personales (PCs)</b> como la Apple I y la IBM PC. Avance en software con <b>sistemas operativos gráficos (Windows, macOS, Linux)</b>.</p>	
<b>Quinta generación y el futuro</b>	<p><b>Computación en la nube, inteligencia artificial, computación cuántica</b> y miniaturización extrema. Dispositivos como <b>smartphones y supercomputadoras</b> muestran la evolución hacia una informática más accesible y poderosa.</p>	

# 02. ARQUITECTURA DE LA COMPUTADORA

# *¿Qué es la arquitectura de las computadoras?*

La arquitectura de computadoras es el conjunto de principios, diseño y estructura que define el funcionamiento interno de un sistema de cómputo. Incluye tanto los componentes físicos (hardware) como la forma en que estos interactúan con el software para procesar información de manera eficiente.



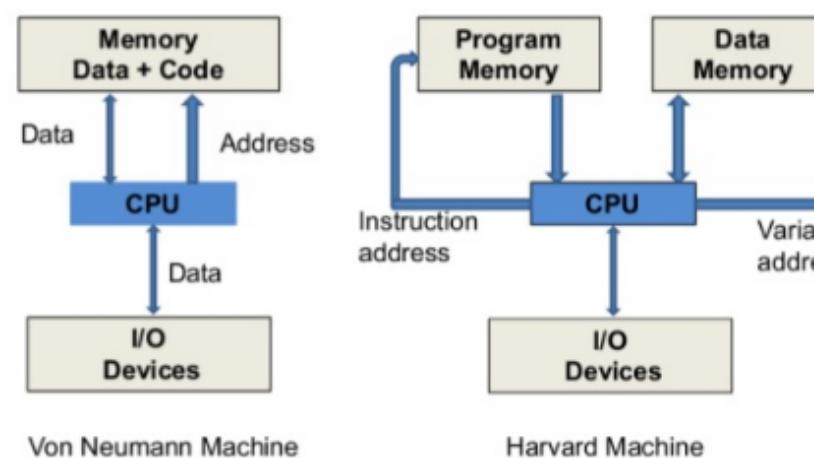
# *Componentes principales de la arquitectura del computador*

CPU	BUS	Memoria principal	E/S
<p>Es la unidad central de procesamiento o microprocesador. Esta parte se encarga de ir ejecutando las diferentes instrucciones de la ISA y los datos que el software emplea para su ejecución. Es decir, es la encargada de ejecutar los programas informáticos, incluido el sistema operativo.</p>	<p>Se refiere a los componentes que entrelazan partes de la computadora y pueden ser de varios tipos y características, como el bus de datos, el bus de direcciones, y el bus de control.</p>	<p>Es la memoria RAM, generalmente, donde se guardan los programas que se van a ejecutar, es decir, los datos e instrucciones necesarios para un proceso y que serán reclamados por la CPU.</p>	<p>Los dispositivos de entrada y salida permiten la interacción entre el computador y el usuario. Estos dispositivos incluyen teclados, pantallas, impresoras, escáneres, entre otros.</p>

# TIPOS DE ARQUITECTURAS

von Neuman	Arquitectura Harvard	RISC	CISC
<p>Es el modelo clásico de arquitectura de computadoras, donde tanto los datos como las instrucciones se almacenan en la misma memoria y se procesan secuencialmente. Tiene una unidad de control y una unidad aritmético lógica (ALU) para ejecutar instrucciones.</p>	<p>Similar al modelo de von Neuman, pero con memoria separada para datos e instrucciones. Esto permite la recuperación de instrucciones y datos simultáneamente, mejorando la eficiencia en ciertos casos.</p>	<p>Se caracteriza por un conjunto de instrucciones más pequeñas y simples, lo que facilita la decodificación y ejecución más rápida de instrucciones. Suelen tener ciclos de instrucción más cortos.</p>	<p>En contraste con RISC, las computadoras CISC tienen un conjunto de instrucciones más amplio y complejo. Cada instrucción CISC puede realizar múltiples operaciones, lo que puede reducir la cantidad total de instrucciones necesarias para un programa.</p>

## Tipos de arquitectura de computadoras

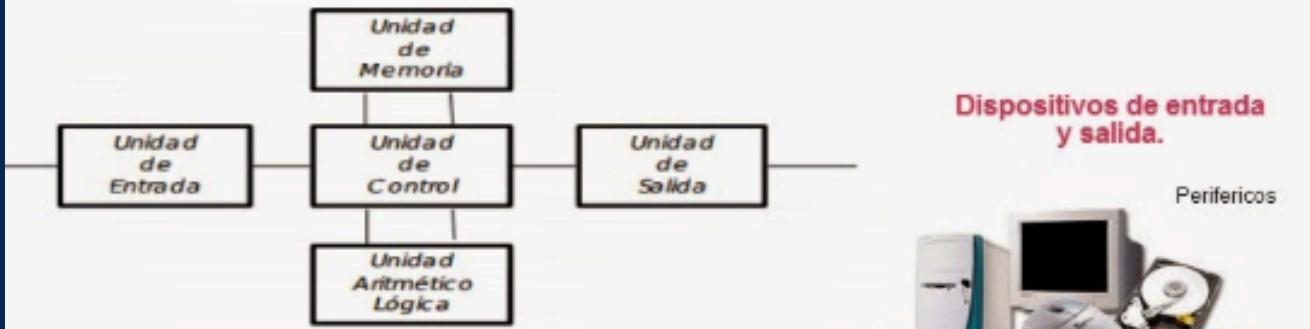


Isaac. (2022, 26 de abril). Tipos de arquitectura de un ordenador. Profesional Review. Recuperado de <https://www.profesionalreview.com/2022/04/26/tipos-de-arquitectura-de-un-ordenador/>

# Arquitectura de las computadoras



Es el diseño conceptual y la estructura operacional fundamental de un sistema de computadora.



## MEMORIAS: RAM-ROM



La ROM  
Es de sólo lectura.



Perifericos



MEMORIA RAM  
Se utiliza como memoria de trabajo para el sistema operativo, los programas y la mayor parte del software.

## HARDWARE Y SOFTWARE

Hardware: Parte física de la computadora.



Software: Formado por programas necesarios.  
4 diferentes Categorías: Sistemas Operativos, Lenguajes de Programación, Software de Aplicación.



## SISTEMA OPERATIVO

Supervisa y controla todas las actividades.



# 03. REPRESENTACIÓN DE LOS DATOS

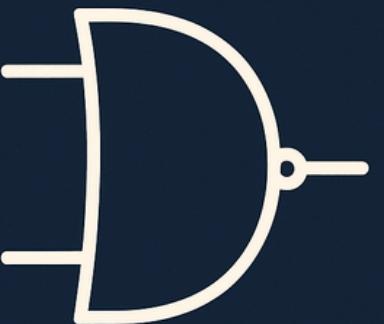
Los datos en una computadora se representan mediante sistemas numéricos y códigos específicos. Esta representación es fundamental para el procesamiento y almacenamiento de la información:

## Sistema Binario

Es la base del funcionamiento de las computadoras, ya que trabajan con bits (0 y 1) para representar datos y ejecutar operaciones lógicas y aritméticas.

### SISTEMA BINARIO

10110  
01101  
11010  
10011  
10011



## Sistemas de Codificación

Permiten representar caracteres, símbolos y otros datos en formato legible para los humanos. Ejemplos comunes son ASCII y Unicode, que asignan valores numéricos a cada carácter.

## SISTEMAS DE CODIFICACIÓN

ASCII

65 A

UNICODE

0041 A

## Representación de Números

Además del binario, se usan otros sistemas como el hexadecimal y el octal, que simplifican la lectura y manipulación de datos en bajo nivel.

### REPRESENTACIÓN DE NÚMEROS

BINARIO **1010**

HEXADECIMAL **1F**

OCTAL **52**

## Operaciones Lógicas y Aritméticas

Las computadoras utilizan puertas lógicas para realizar operaciones como AND, OR, XOR y NOT, fundamentales para la computación digital.

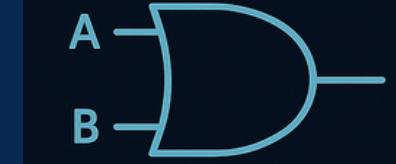
### AND



### OR

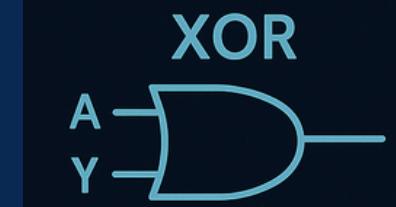
TRUTH TABLE			OUTPUT
A	B	C	0 0
0	0	0	0 0
1	1	1	1 1
1	0	1	1 1

### XOR



### NOT

TRUTH TABLE		A
0	1	0
1	1	0
1	0	1



# 04. INTERFAZ HOMBRE COMPUTADOR

La interacción entre los humanos y las computadoras es clave para la usabilidad y accesibilidad de los sistemas. El diseño de interfaces efectivas impacta directamente en la experiencia del usuario y la productividad.

### **ASPECTOS IMPORTANTES:**

#### **Interfaces Gráficas de Usuario (GUI)**

Estas interfaces permiten la interacción visual con el sistema a través de elementos como ventanas, iconos y botones. La evolución de las GUI ha mejorado la accesibilidad y facilidad de uso.

#### **Dispositivos de Entrada**

Incluyen teclados, ratones, pantallas táctiles, comandos de voz y gestos, cada uno con ventajas según el contexto de uso.

#### **Principios de Usabilidad**

una buena interfaz debe ser intuitiva, ofrecer retroalimentación clara y minimizar la carga cognitiva del usuario.

#### **Accesibilidad**

El diseño inclusivo permite que personas con discapacidades puedan interactuar con los sistemas computacionales mediante tecnologías como lectores de pantalla, interfaces hapticas y control por voz.

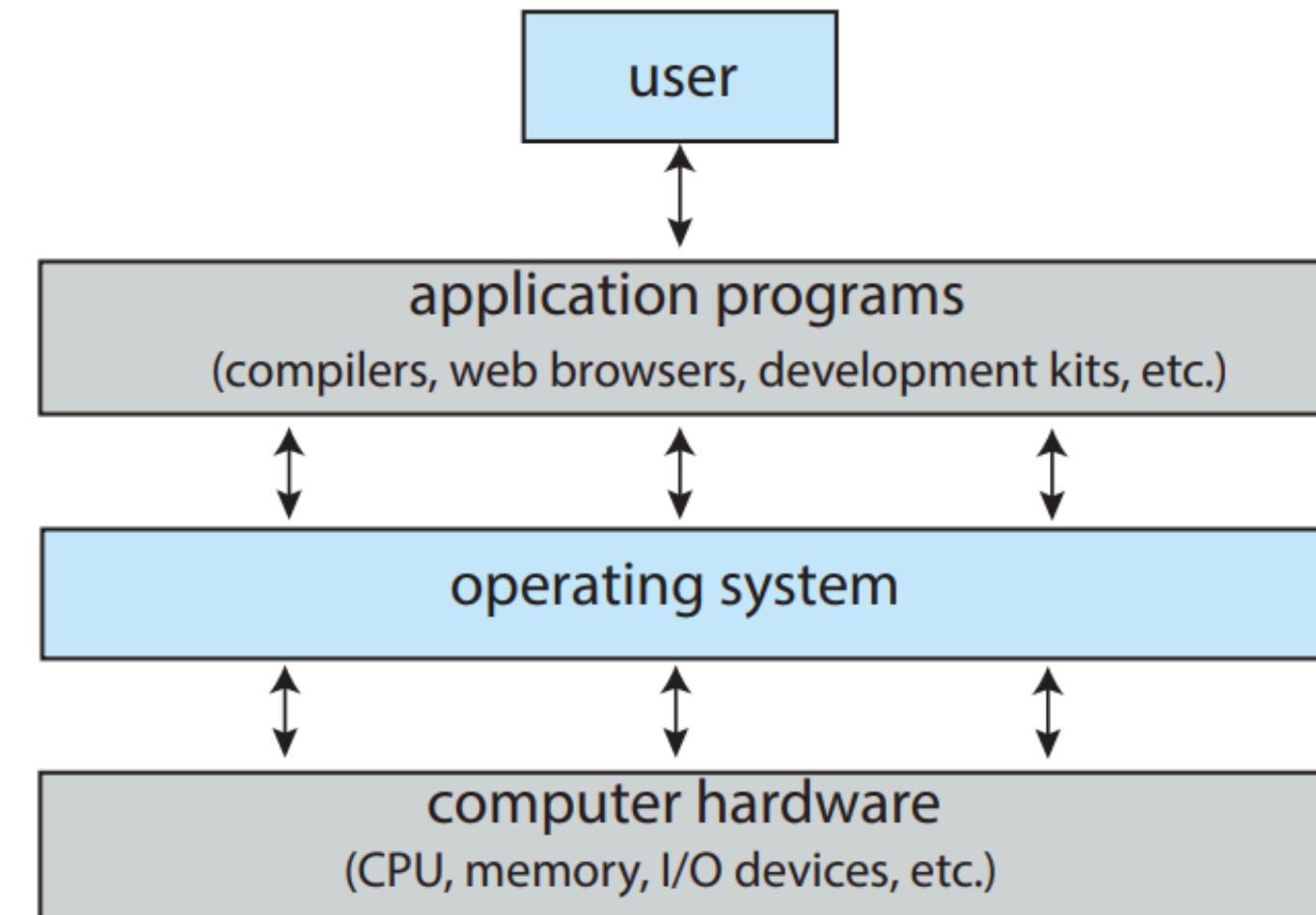
#### **Experiencia del Usuario**

La percepción y satisfacción del usuario con la interfaz depende del diseño, eficiencia y facilidad de uso del sistema.

# 05. SISTEMAS OPERATIVOS

## 05. SISTEMAS OPERATIVOS

- Un Sistema Operativo (SO) es el software fundamental que actúa como intermediario entre el hardware de una computadora y los programas que el usuario desea ejecutar.
- Su función principal es gestionar los recursos del sistema (procesador, memoria, almacenamiento, dispositivos de entrada/salida) y proporcionar una interfaz para que los usuarios y las aplicaciones puedan interactuar con el hardware de manera eficiente y segura.



**Figure 1.1** Abstract view of the components of a computer system.

## 05. SISTEMAS OPERATIVOS

- Ejemplo práctico:

Cuando un programa en Python abre un archivo con:

```
open('archivo.txt', 'r')
```

```
# Abrir el archivo en modo lectura ('r')
with open('archivo.txt', 'r', encoding='utf-8') as archivo:
    contenido = archivo.read() # Leer el contenido completo del archivo

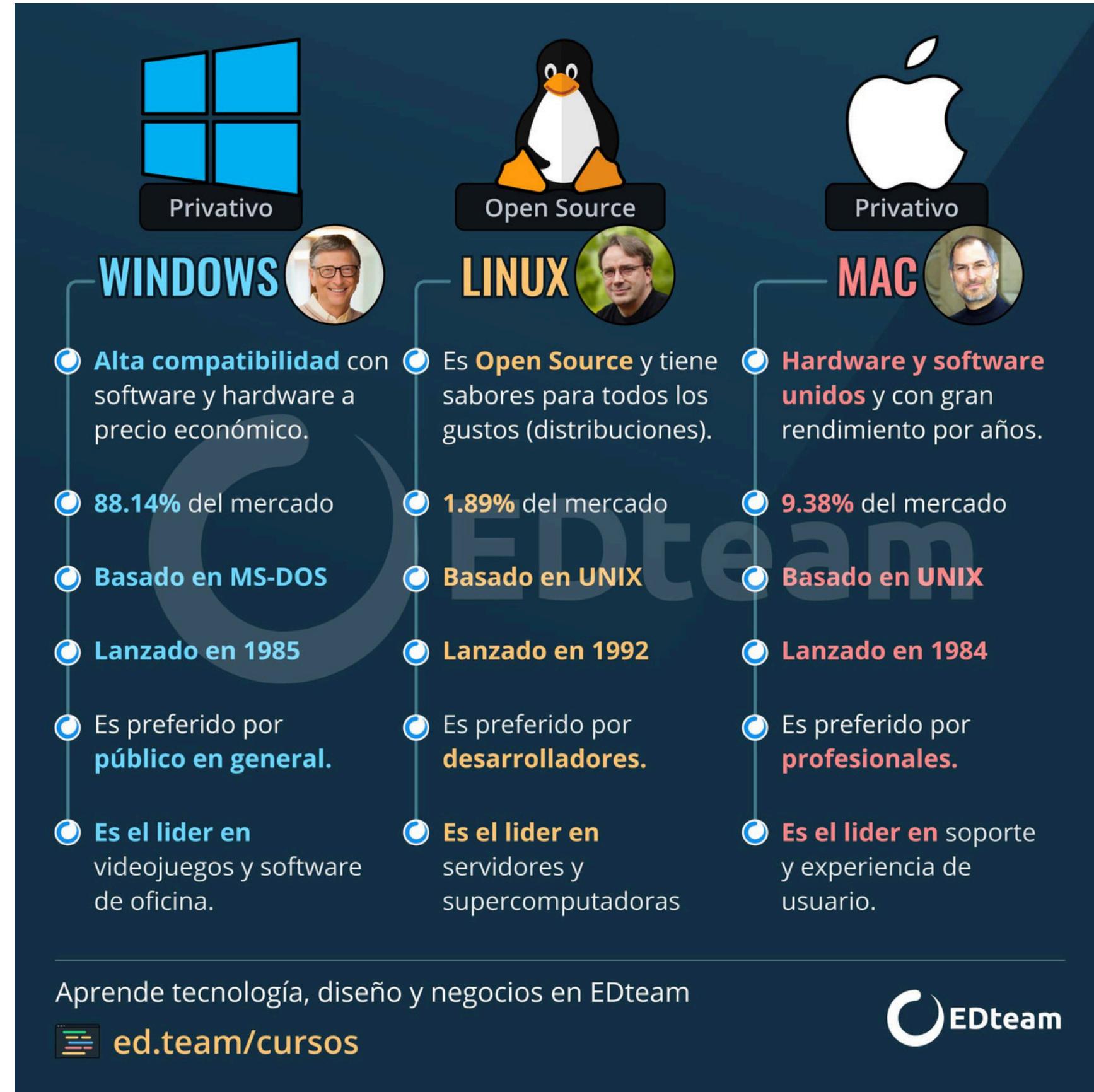
# Mostrar el contenido en pantalla
print(contenido)
```

El SO se encarga de localizar el archivo, asignar memoria y permitir la lectura segura de datos.

## 05. SISTEMAS OPERATIVOS



# 05. SISTEMAS OPERATIVOS



# 06. COMPILADORES Y TRADUCTORES

## 06. COMPILADORES Y TRADUCTORES

- Un **compilador** es un **tipo de traductor** diseñado para **convertir código escrito en un lenguaje de programación de alto nivel**, que es fácilmente comprensible por los programadores, **a un lenguaje de bajo nivel**, comprensible por las computadoras.
- Algunos compiladores no producen directamente código máquina, sino que generan un **programa en lenguaje ensamblador**. En estos casos, es necesario utilizar un **ensamblador adicional** para convertir este programa en código máquina.

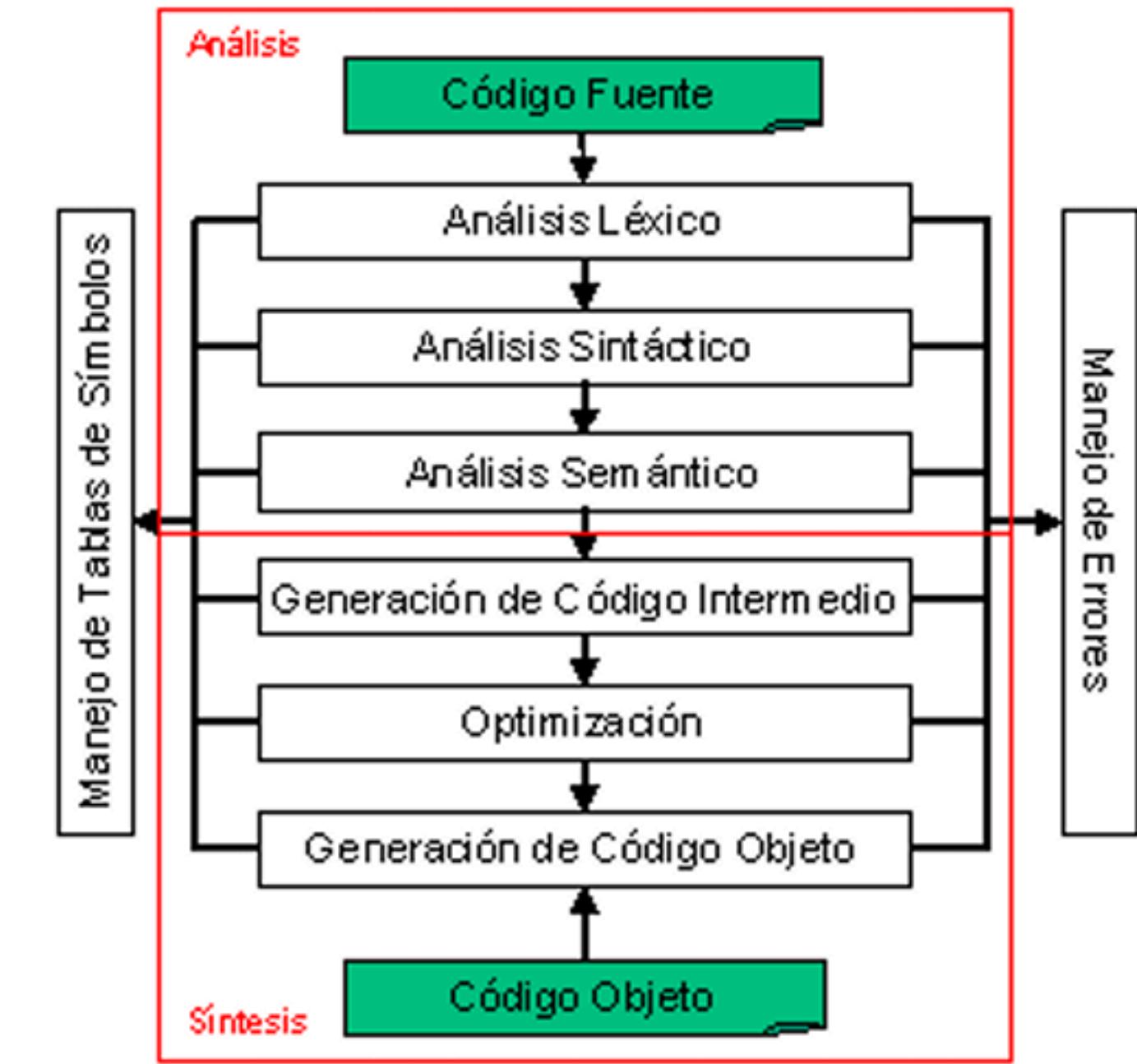


Figura 1. Estructura de un compilador.

## 06. COMPILEDORES Y TRADUCTORES

- Un **traductor** es un término más general que incluye a los **compiladores, intérpretes y ensambladores**.
- Los **intérpretes** son programas que leen y ejecutan **instrucciones de un código fuente una por una**. Durante este proceso, **coexisten en memoria con el programa fuente**, llevando a cabo la **traducción en tiempo real mientras se ejecuta el programa**. Generalmente, un intérprete lleva a cabo dos operaciones:

Primero, traduce el código fuente a un formato **intermedio** y, luego, interpreta este código traducido. Si la transformación produce un formato intermedio, este suele ser el resultado de un análisis sintáctico semántico, similar a una compilación a código intermedio.

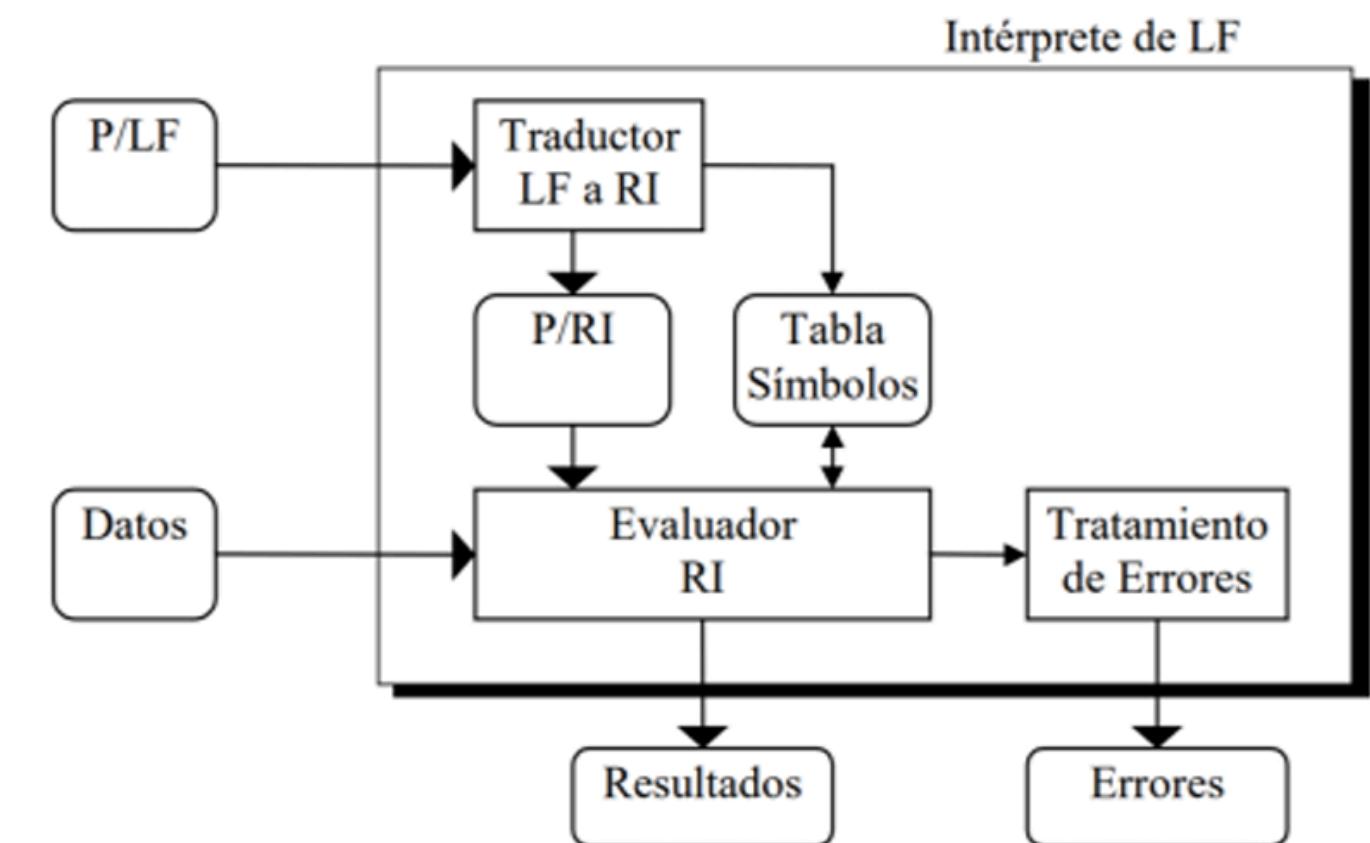
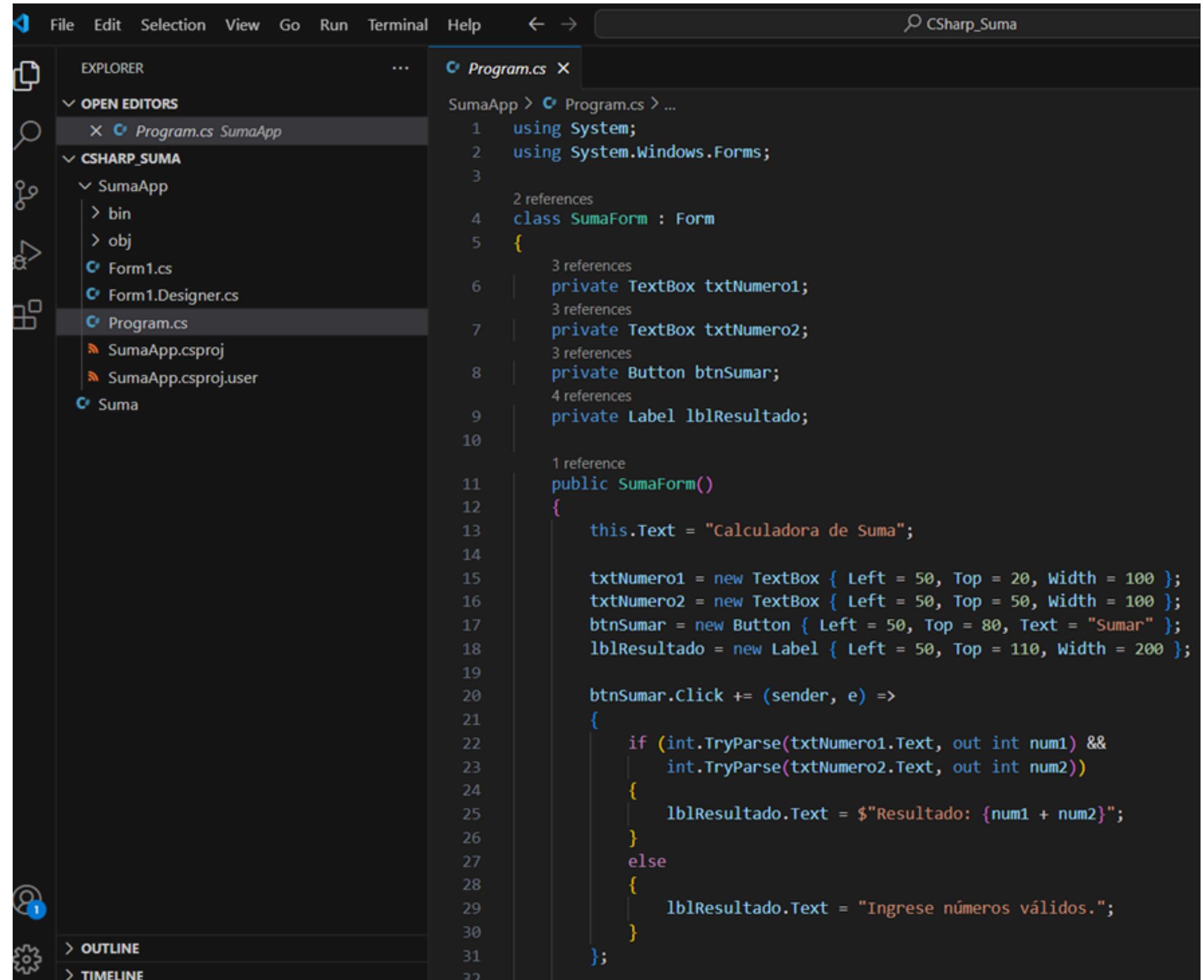


Figura 2. Estructura de un intérprete.

# 06. COMPILADORES Y TRADUCTORES

- Demostración de como se compila y ejecuta una aplicación de interfaz gráfica en un lenguaje compilado como C#



```
File Edit Selection View Go Run Terminal Help ← → CSharp_Suma

EXPLORER
OPEN EDITORS
Program.cs SumaApp
CSHARP_SUMA
SumaApp
bin
obj
Form1.cs
Form1.Designer.cs
Program.cs
SumaApp.csproj
SumaApp.csproj.user
Suma

Program.cs X
SumaApp > Program.cs > ...
1 using System;
2 using System.Windows.Forms;
3
4 class SumaForm : Form
5 {
6     private TextBox txtNumero1;
7     private TextBox txtNumero2;
8     private Button btnSumar;
9     private Label lblResultado;
10
11    public SumaForm()
12    {
13        this.Text = "Calculadora de Suma";
14
15        txtNumero1 = new TextBox { Left = 50, Top = 20, Width = 100 };
16        txtNumero2 = new TextBox { Left = 50, Top = 50, Width = 100 };
17        btnSumar = new Button { Left = 50, Top = 80, Text = "Sumar" };
18        lblResultado = new Label { Left = 50, Top = 110, Width = 200 };
19
20        btnSumar.Click += (sender, e) =>
21        {
22            if (int.TryParse(txtNumero1.Text, out int num1) &&
23                int.TryParse(txtNumero2.Text, out int num2))
24            {
25                lblResultado.Text = $"Resultado: {num1 + num2}";
26            }
27            else
28            {
29                lblResultado.Text = "Ingrese números válidos.";
30            }
31        };
32    }
}
```

# 06. COMPILADORES Y TRADUCTORES

The screenshot shows a terminal window with the following command history:

```
PS D:\Demosvsc\CSharp_Suma> cd SumaApp
PS D:\Demosvsc\CSharp_Suma\SumaApp> dotnet build
Restauración completada (0.2s)
    SumaApp realizado correctamente (4.4s) → bin\Debug\net9.0-windows\SumaApp.dll

    Compilación realizado correctamente en 5.0s
PS D:\Demosvsc\CSharp_Suma\SumaApp> dotnet run
PS D:\Demosvsc\CSharp_Suma\SumaApp> dotnet run
PS D:\Demosvsc\CSharp_Suma\SumaApp> dotnet publish -c Release -r win-x64 --self-contained true
Restauración completada (5.4s)
    SumaApp realizado correctamente (3.0s) → bin\Release\net9.0-windows\win-x64\publish\

    Compilación realizado correctamente en 8.8s
PS D:\Demosvsc\CSharp_Suma\SumaApp>
```

A red box highlights the last line of the terminal output: "Compilación **realizado correctamente** en 8.8s".

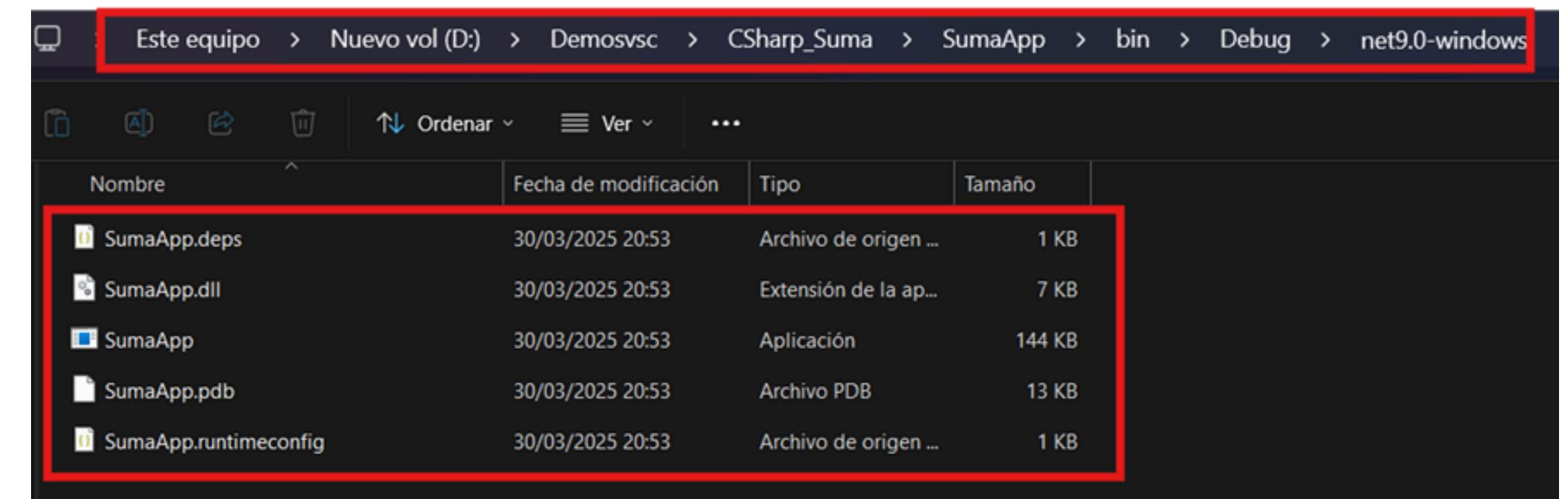
# 06. COMPILADORES Y TRADUCTORES

The screenshot shows a development environment with the following components:

- Code Editor:** Displays the `Program.cs` file for a Windows application named `SumaApp`. The code defines a `SumaForm` class with properties for two text boxes (`txtNumero1`, `txtNumero2`), a button (`btnSumar`), and a label (`lblResultado`). It also contains a constructor that sets the form's text to "Calculadora de Suma".
- Terminal:** Shows the command `dotnet run` being executed in the directory `D:\Demosvsc\CSharp_Suma\SumaApp`. The terminal output indicates a successful build ("realizado correctamente") and the execution of the application.
- Application Window:** A window titled "Calculadora de Suma" is displayed. It has two text input fields containing "10" and "30", a button labeled "Sumar", and a label below it stating "Resultado: 40".

## 06. COMPILADORES Y TRADUCTORES

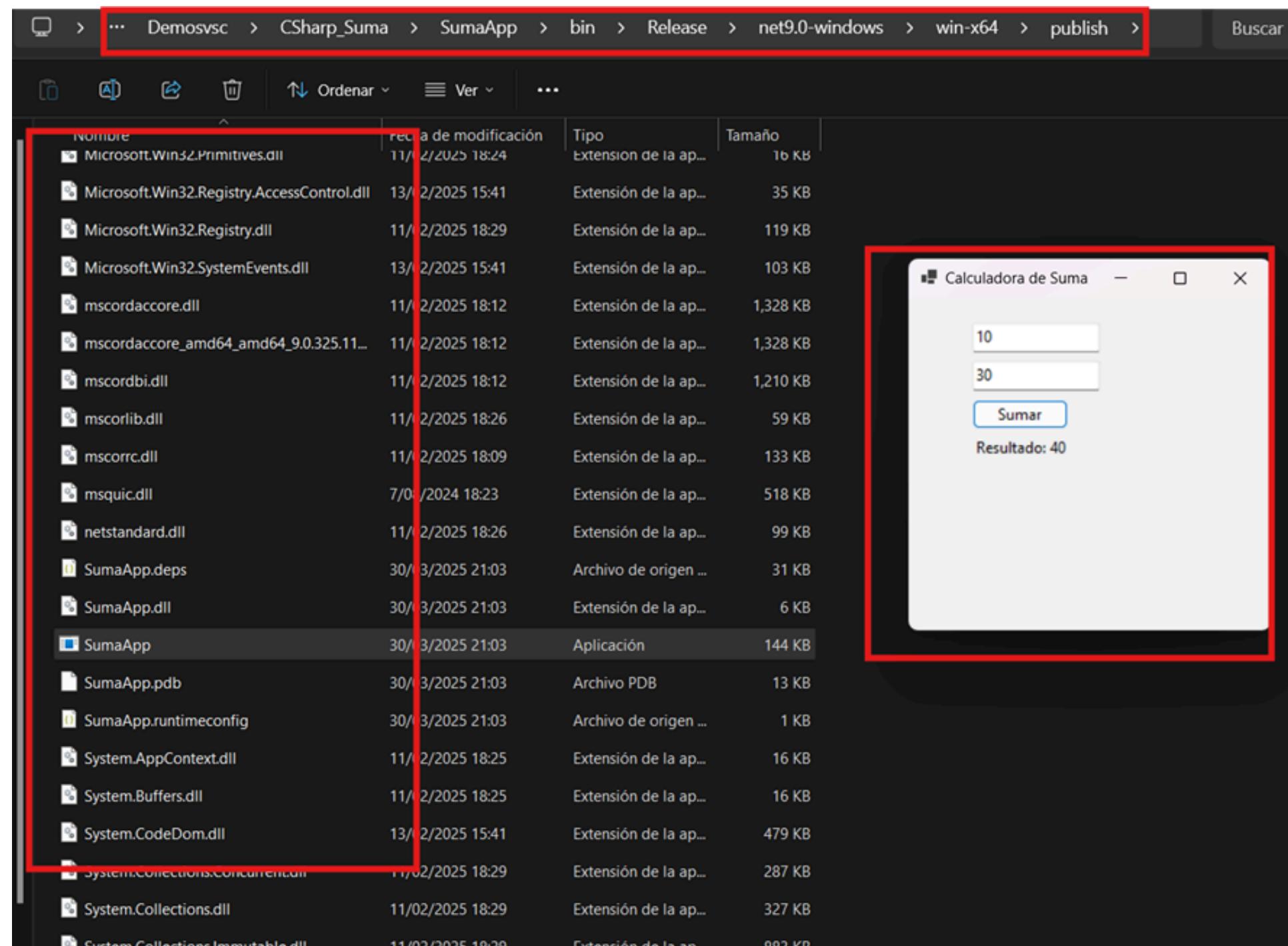
- Genera el **compilado**, con sus ddls y archivos correspondientes



## 06. COMPILADORES Y TRADUCTORES

- Generamos un ejecutable, para ejecutarlo en otro lugar sin Visual Studio Code para lo cual lo cerramos.

Genera el siguiente archivo ejecutable lo cual podemos copiar ese archivo y ejecutarlo en otra PC con Windows.



# 06. COMPILEDORES Y TRADUCTORES

- Programa interpretado con python:

Diferencias clave con C#:

- No se necesita compilar, el código se ejecuta directamente.
- No se genera un ejecutable, siempre necesitas Python instalado.

Nombre	Fecha de modificación
SumaApp	30/03/2025 20:51
CSharp_Suma.sln	30/03/2025 21:44
Suma	30/03/2025 20:32
suma	30/03/2025 21:57

The screenshot shows a Windows desktop environment. On the left, there's a file explorer window with several items listed. In the center, a code editor (Visual Studio Code) displays a Python script named 'suma.py'. The script uses the Tkinter library to create a simple application for adding two numbers. On the right, a terminal window shows the command 'python suma.py' being run, and the output shows the application's window. The application window has a title 'Suma de Dos...', two entry fields for 'Número 1:' and 'Número 2:', and a result field showing 'Resultado: 40.0' with a 'Sumar' button.

```
File Edit Selection View Go Run Terminal Help ← → 🔍 CSharp_Suma
EXPLORER
OPEN EDITORS
  suma.py
CSHARP_SUMA
  SumaApp
  CSharp_Suma.sln
  Suma
  suma.py
suma.py > ...
1 import tkinter as tk
2 from tkinter import messagebox
3
4 def sumar():
5     try:
6         num1 = float(entry1.get())
7         num2 = float(entry2.get())
8         resultado.set(f"Resultado: {num1 + num2}")
9     except ValueError:
10         messagebox.showerror("Error", "Ingrese números válidos")
11
12 # Crear ventana
13 ventana = tk.Tk()
14 ventana.title("Suma de Dos Números")
15
16 tk.Label(ventana, text="Número 1:").pack()
17 entry1 = tk.Entry(ventana)
18 entry1.pack()
19
20 tk.Label(ventana, text="Número 2:").pack()
21 entry2 = tk.Entry(ventana)
22 entry2.pack()
23
24 resultado = tk.StringVar()
25 tk.Label(ventana, textvariable=resultado).pack()
26
27 tk.Button(ventana, text="Sumar", command=sumar).pack()
28
29 ventana.mainloop()
30
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
JLINE
MELINE
Python 3.13.2
PS D:\Demosvsc\CSHARP_SUMA> python suma.py
```

# 06. COMPILADORES Y TRADUCTORES

- Comparación de la demostración hecha

Característica	C# (Compilado)	Python (Interpretado)
Tipo de lenguaje	Compilado	Interpretado
Proceso de ejecución	Compilado a código máquina antes de ejecutarse	Cada línea del código se traduce y ejecuta en tiempo real
Ejemplo de código	Aplicación de suma con Windows Forms	Script de suma en la consola
Generación de ejecutable	Genera un .exe que se ejecuta sin necesidad del código fuente	No genera un ejecutable directamente, requiere el intérprete de Python
Portabilidad	Limitado a plataformas compatibles con .NET	Altamente portable entre sistemas operativos
Velocidad de ejecución	Más rápida debido a la compilación previa	Más lenta debido a la interpretación línea por línea

## 06. COMPILADORES Y TRADUCTORES



# REFERENCIAS BIBLIOGRÁFICAS

SILBERSCHATZ, A., GALVIN, P. B., & GAGNE, G. (2018). OPERATING SYSTEM CONCEPTS (10TH ED.). WILEY.

LIMA Y LIMA, C. M. (2024). COMPILADOR VS INTÉRPRETE: ¿AL MEJOR ELECCIÓN AL TRADUCIR UN LENGUAJE? UNIVERSIDAD DE SAN CARLOS DE GUATEMALA.

STALLINGS, W. (2017). COMPUTER ORGANIZATION AND ARCHITECTURE (11TH ED.). PEARSON.

TANENBAUM, A. S., & AUSTIN, T. (2013). STRUCTURED COMPUTER ORGANIZATION (6TH ED.). PEARSON.

MANO, M. M., & KIME, C. R. (2017). LOGIC AND COMPUTER DESIGN FUNDAMENTALS (5TH ED.). PEARSON.

NORMAN, D. A. (2013). THE DESIGN OF EVERYDAY THINGS (REVISED ED.). BASIC BOOKS.

SHNEIDERMAN, B., PLAISANT, C., COHEN, M., & JACOBS, S. (2017). DESIGNING THE USER INTERFACE: STRATEGIES FOR EFFECTIVE HUMAN-COMPUTER INTERACTION (6TH ED.). PEARSON

DIX, A., FINLAY, J., ABOWD, G., & BEALE, R. (2004). HUMAN-COMPUTER INTERACTION (3RD ED.). PEARSON.

# REFERENCIAS BIBLIOGRÁFICAS

PROFESIONAL REVIEW. (2022). ARQUITECTURA DE COMPUTADORAS. RECUPERADO DE: [HTTPS://WWW.PROFESIONALREVIEW.COM/2022/10/01/ARQUITECTURA-DE-COMPUTADORAS/](https://www.profesionalreview.com/2022/10/01/arquitectura-de-computadoras/)

ISAAC. (2022, 26 DE ABRIL). TIPOS DE ARQUITECTURA DE UN ORDENADOR. PROFESIONAL REVIEW. RECUPERADO DE [HTTPS://WWW.PROFESIONALREVIEW.COM/2022/04/26/TIPOS-DE-ARQUITECTURA-DE-UN-ORDENADOR/](https://www.profesionalreview.com/2022/04/26/tipos-de-arquitectura-de-un-ordenador/)

MUCHAS GRACIAS

```
    render() {
      return (
        <React.Fragment>
          <div className="py-5">
            <div className="container">
              <Title name="our" title="product">
              <div className="row">
                <ProductConsumer>
                  {(value) => {
                    |   |   |   console.log(value)
                  }}
                </ProductConsumer>
              </div>
            </div>
          </div>
        <React.Fragment>
```