UNIVERSIDAD AUTÓNOMA "TOMAS FRÍAS" CARRERA DE INGENIERÍA DE SISTEMAS				
Materia:	Arquitectura de computadoras (SIS-522)			
Estudiante:	Univ. Jonatan Porco Jaita			N° Práctica
Docente:	Ing. Gustavo A. Puita Choque			
Auxiliar:	Univ. Aldrin Roger Perez Miranda			9
06/11/2024	**Fecha publicación**			
20/11/2024	**Fecha de entrega**			
Grupo:	**1**	**Sede**	**Potosí**	

1) ¿Qué es el 'stack' en el contexto del lenguaje ensamblador y cómo se utiliza? ***(10 pts)***

En el contexto del lenguaje ensamblador, el stack es una estructura de datos de tipo LIFO (Last In, First Out), es decir, el último elemento que se almacena en el stack es el primero en ser retirado. Se utiliza principalmente para almacenar temporalmente datos, direcciones de retorno, parámetros de funciones o registros, mientras el programa realiza operaciones.
Utilización:

Se accede al stack mediante los registros SP (Stack Pointer) y BP (Base Pointer).

Las instrucciones PUSH y POP permiten agregar o retirar valores del stack.

Ejemplo: Si se realiza un PUSH AX, se almacena el valor del registro AX en el stack, y con un POP BX, se extrae ese valor del stack y se asigna al registro BX.

2) Describe un escenario práctico donde el uso de ensamblador sería más ventajoso que el uso de un lenguaje de alto nivel. (10 pts)
Un escenario práctico donde el uso de ensamblador sería más ventajoso que el de un lenguaje de alto nivel es en el desarrollo de controladores de hardware o firmware embebido. Por ejemplo, en sistemas embebidos, donde se necesita interactuar directamente con el hardware, controlar registros específicos o realizar operaciones críticas en tiempo real.
Razón:
El ensamblador permite un control absoluto sobre el hardware, optimizando el uso de recursos y tiempo de ejecución, algo que los lenguajes de alto nivel no logran debido a la abstracción que introducen. Por ejemplo, al programar el controlador de un microcontrolador en un sistema de control industrial, el ensamblador permite interactuar directamente con los registros del dispositivo, asegurando un rendimiento óptimo.

"describir"

3) Explique cada línea del siguiente código del lenguaje ensamblador y diga que es lo que se está haciendo ***(20 pts)***

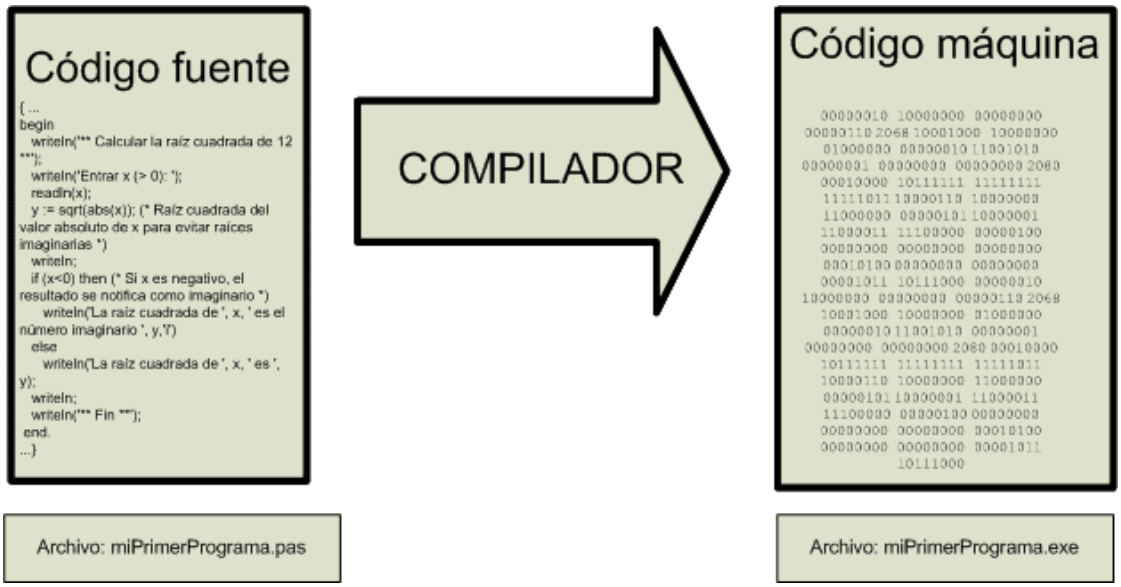
```
MOV AX, 5       ; Línea 1
MOV BX, 10      ; Línea 2
ADD AX, BX      ; Línea 3
MOV CX, AX      ; Línea 4
```

Explicación del código

1. Línea 1: MOV AX, 5
   Se mueve el valor 5 al registro AX.
   Lo que se está haciendo: Se inicializa el registro AX con el valor 5.
2. Línea 2: MOV BX, 10
   Se mueve el valor 10 al registro BX.
   Lo que se está haciendo: Se inicializa el registro BX con el valor 10.
3. Línea 3: ADD AX, BX
   Se suma el contenido de los registros AX y BX, y el resultado se almacena en AX.
   Lo que se está haciendo: Se realiza la operación AX = AX + BX, es decir, AX = 5 + 10 = 15.
4. Línea 4: MOV CX, AX
   Se mueve el valor contenido en AX (que ahora es 15) al registro CX.
   Lo que se está haciendo: Se transfiere el resultado de la suma al registro CX para un uso ***posterior.***

   El programa realiza una suma de los valores 5 y 10, almacenados en los registros AX y BX, respectivamente, y guarda el resultado (15) en el registro CX.

4) Explique detalladamente cómo funcionan los compiladores***(10 pts)***



Un compilador es un programa que traduce el **código fuente** escrito en un lenguaje de programación de alto nivel (como Pascal, C++ o Python) a **código máquina**, que puede ser entendido y ejecutado directamente por el procesador de la computadora.

Código fuente :

Es el programa escrito por el programador en un lenguaje de alto nivel (como Pascal en este caso).
En el ejemplo de la imagen, el código fuente contiene instrucciones como calcular la raíz cuadrada de un número, verificar si el número es negativo y trabajar con números imaginarios.

Este archivo está en formato legible para humanos, como miPrimerPrograma.pas.
Compilador

Toma el código fuente y lo **traduce** a código máquina.
Aquí es donde ocurren las etapas (análisis léxico, sintáctico, semántico, generación y optimización de código).
El objetivo del compilador es convertir el código fuente en un archivo ejecutable eficiente.
Código máquina :

El compilador genera un archivo binario (miPrimerPrograma.exe) que contiene instrucciones en código máquina (secuencias de 0s y 1s).
Este código es interpretado directamente por el procesador para ejecutar el programa.
El archivo final no es legible para los humanos, pero es el resultado que la computadora necesita para realizar las operaciones descritas en el código fuente.

Etapas de funcionamiento de compilador:

1. **Análisis léxico:**

- En esta etapa, el compilador divide el código fuente en unidades básicas llamadas **tokens** (palabras clave, operadores, identificadores, etc.).
- **Ejemplo:** Si el código contiene x = x + 1;, se identifican tokens como x, =, +, y 1.

2. **Análisis sintáctico**

- Se valida la **estructura gramatical** del código según las reglas del lenguaje. Si hay errores de sintaxis, se notifican en esta etapa.
- **Ejemplo:** Se verifica que la asignación x = x + 1; tenga la forma correcta según la gramática del lenguaje.

3. **Análisis semántico:**

- El compilador revisa que las operaciones y expresiones tengan sentido lógico. Por ejemplo, asegura que no intentes sumar un número con un texto.
- **Ejemplo:** Si x está definido como número, el compilador comprueba que las operaciones con x sean válidas.

4. **Generación de código intermedio:**

- Se genera un código en un formato intermedio que es más fácil de optimizar y convertir a código máquina.
- **Ejemplo:** Un programa en Pascal podría ser traducido a un formato genérico antes de convertirse en binario.

5. **Optimización del código:**

- El compilador mejora el código intermedio para que sea más eficiente en términos de tiempo de ejecución y uso de recursos.
- **Ejemplo:** Si el programa tiene una operación redundante, el compilador puede simplificarla.

6. **Generación de código máquina:**

- El código optimizado se convierte en instrucciones binarias específicas para el procesador.
- **Ejemplo:** El compilador traduce el código intermedio a instrucciones como MOV y ADD en lenguaje ensamblador, y luego a código binario.
- Técnicamente el compilador actúa como un "traductor" entre el programador y la computadora, asegurándose de que el código fuente sea correctamente interpretado y transformado en un formato que la máquina pueda ejecutar

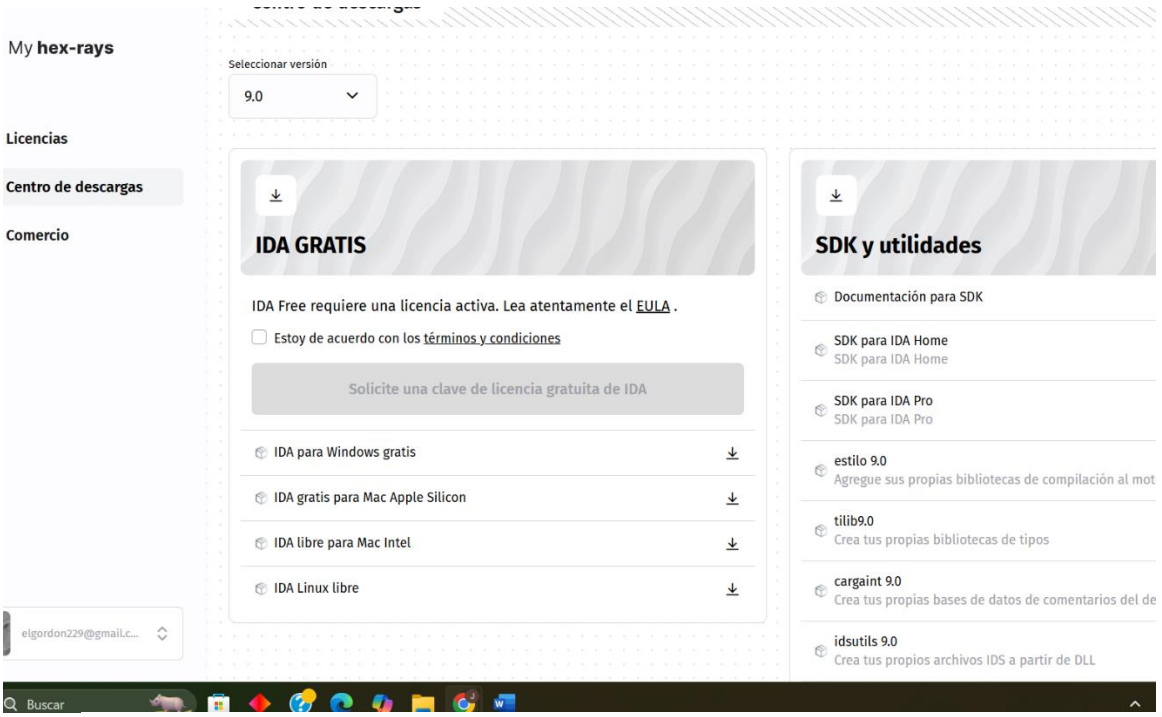
5) Realizar sus propias capturas de pantalla del siguiente procedimiento: ***(50 pts)***

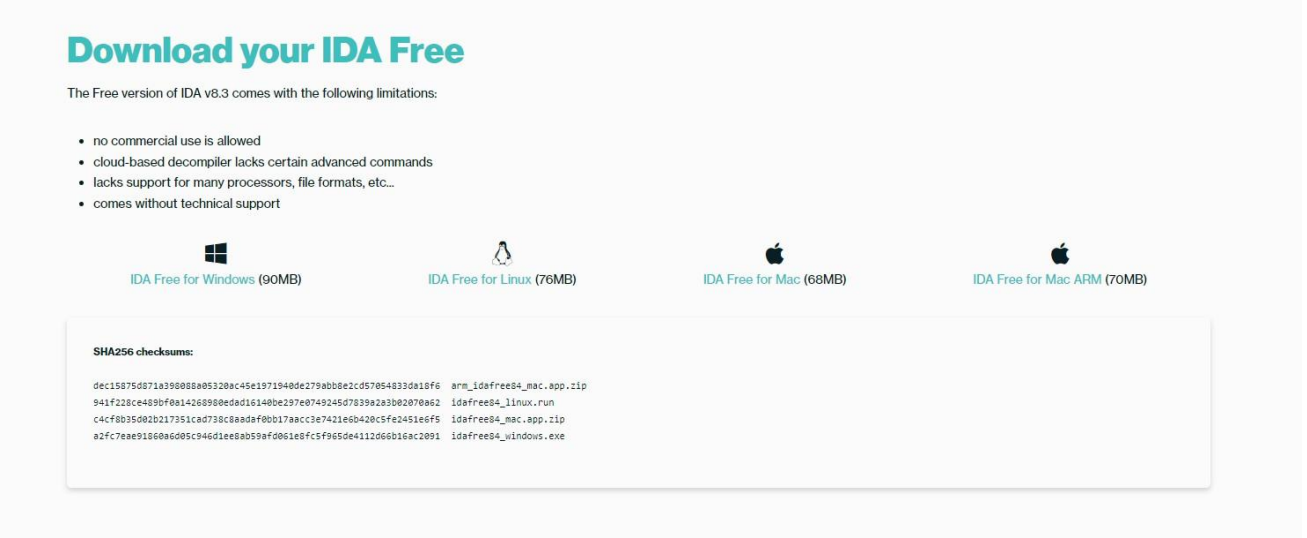
EL PROCEDIMIENTO LO DEBE HACER COMO UN LABORATORIO PASO A PASO Y EXPLICAR QUE ES LO QUE SE ESTA HACIENDO CON SU RESPECTIVA CAPTURA USTED DEBE SELECCIONAR CUALQUIER SERVICIO DE SU PREFERENCIA

IDA: Es una de las herramientas más conocidas y potentes para el análisis de código binario y desensamblado. En este laboratorio se instalará IDA FREE pero también se tiene la versión de paga IDA PRO

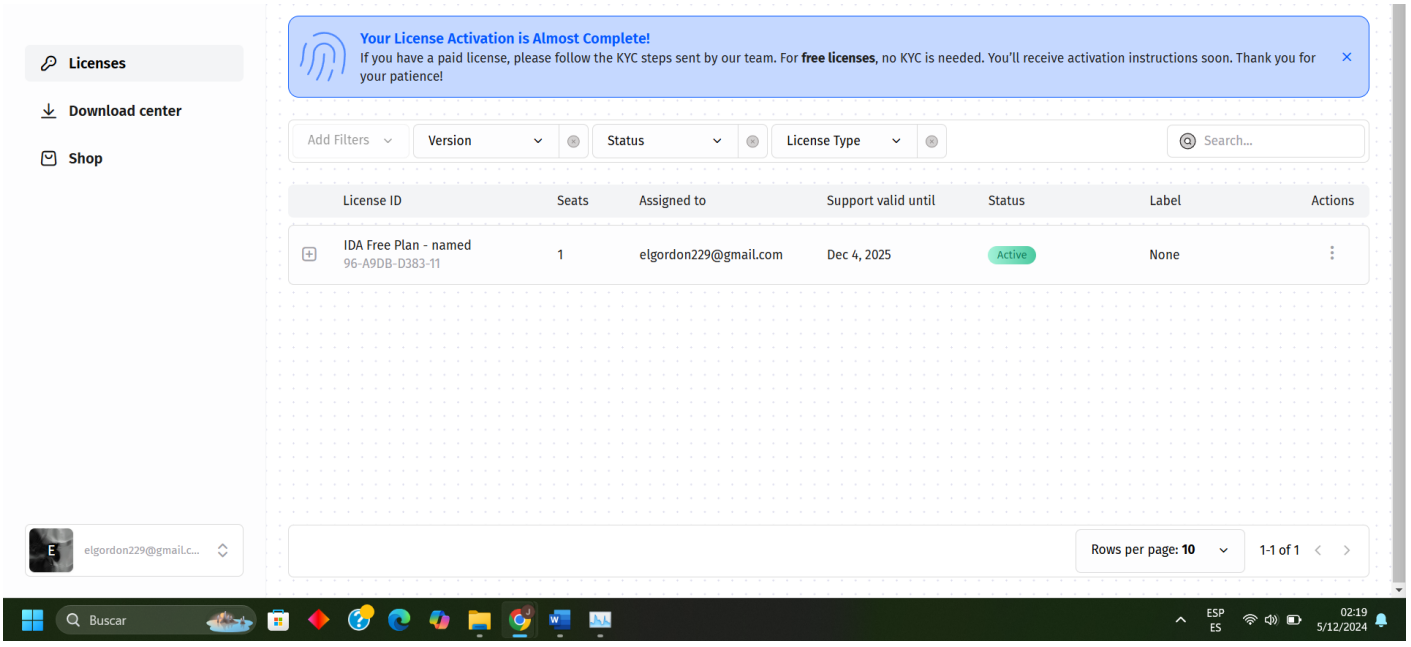
Paso 1:

Descargar el software IDA FREE el cual lo podrá a hacer del siguiente enlace: https://hex-rays.com/ida-free/





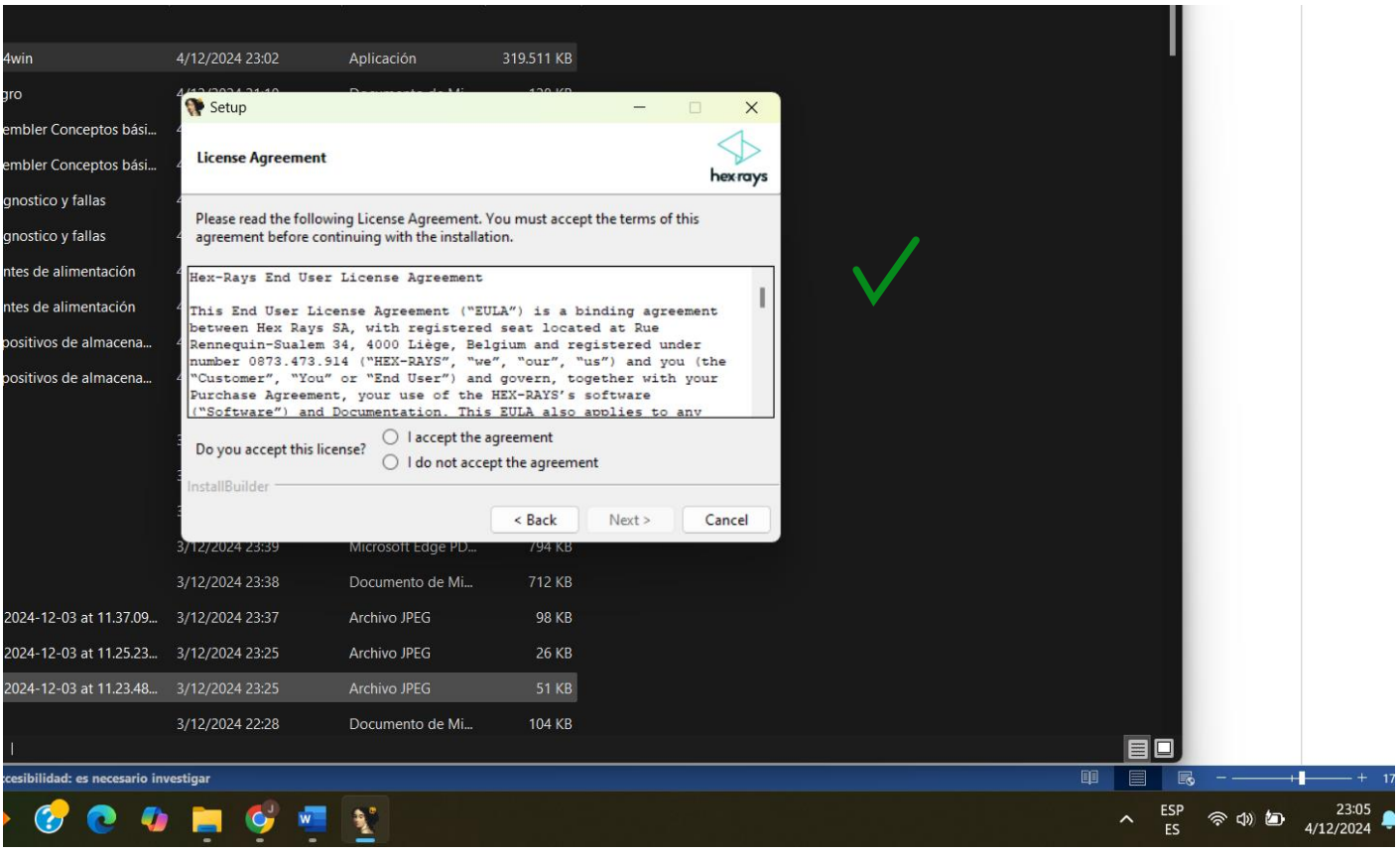
TUVE QUE DESCARGAR LA LICENCIA MAS POR QUE NO ME DEJABA HACER NADA SIN ELLA

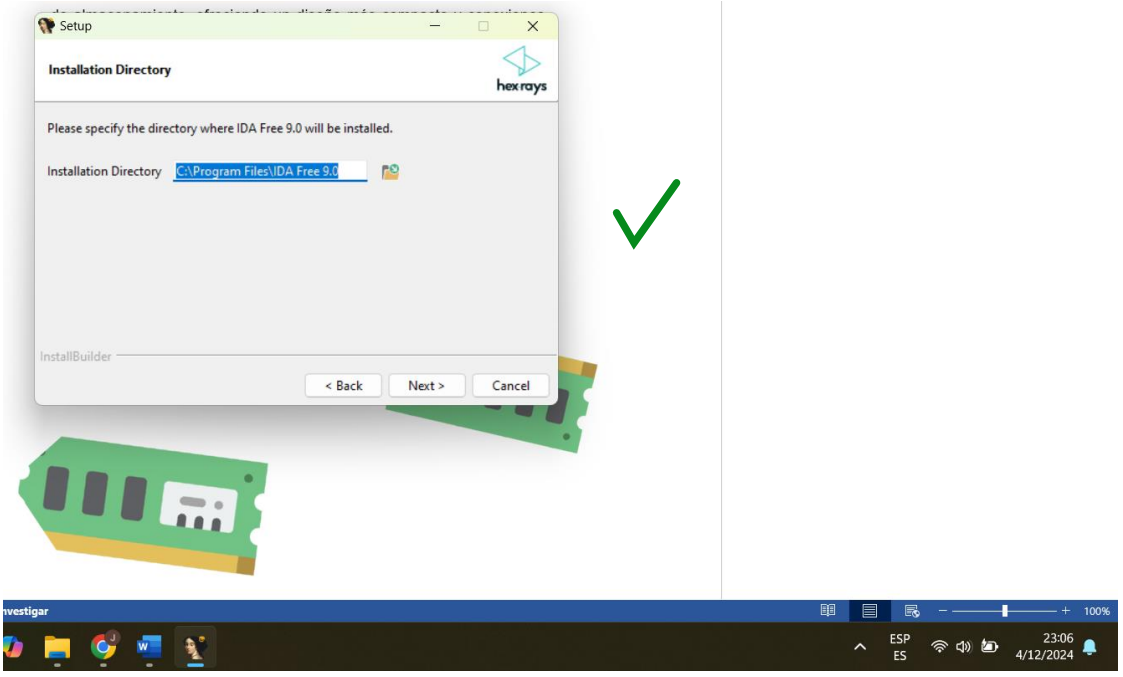


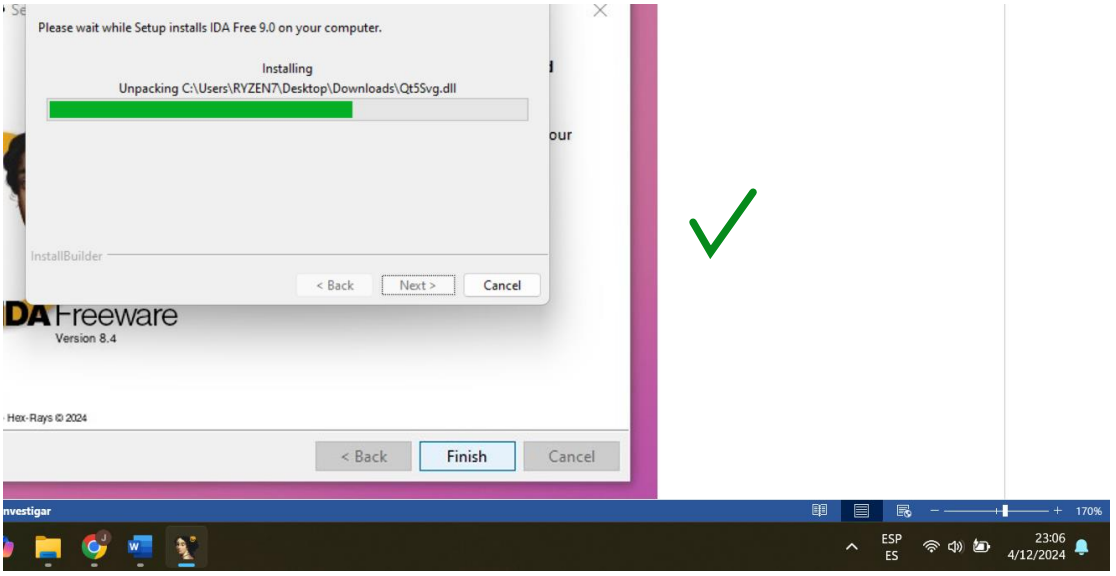
Paso 2:

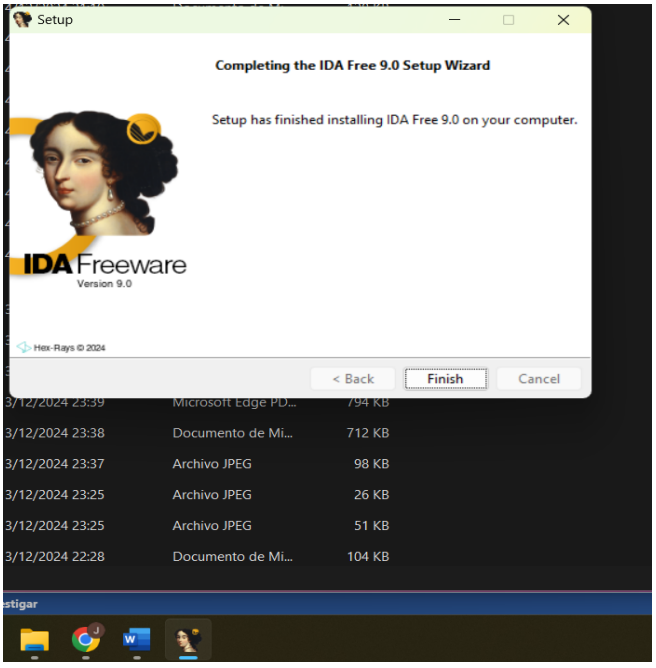
Instalación

Seleccionar Next



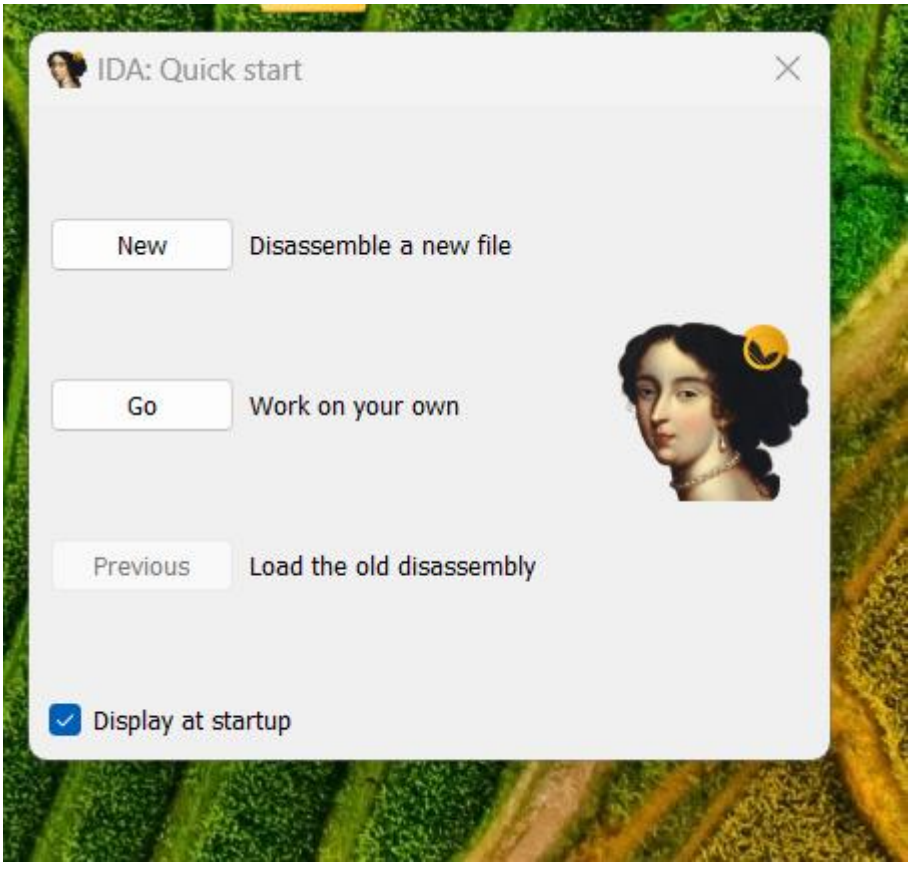

Setup
Installation Directory
hex rays
Please specify the directory where IDA Free 9.0 will be installed.
Installation Directory
C:\Program Files\IDA Free 9.0
InstallBuilder
< Back
Next >
Cancel
100%
ESP
ES
23:06
4/12/2024


Please wait while Setup installs IDA Free 9.0 on your computer.
Installing
Unpacking C:\Users\RYZEN7\Desktop\Downloads\Qt5Svg.dll
InstallBuilder
< Back
Next >
Cancel
Freeware
Version 8.4
Hex-Rays © 2024
< Back
Finish
Cancel
170%
ESP
ES
23:06
4/12/2024



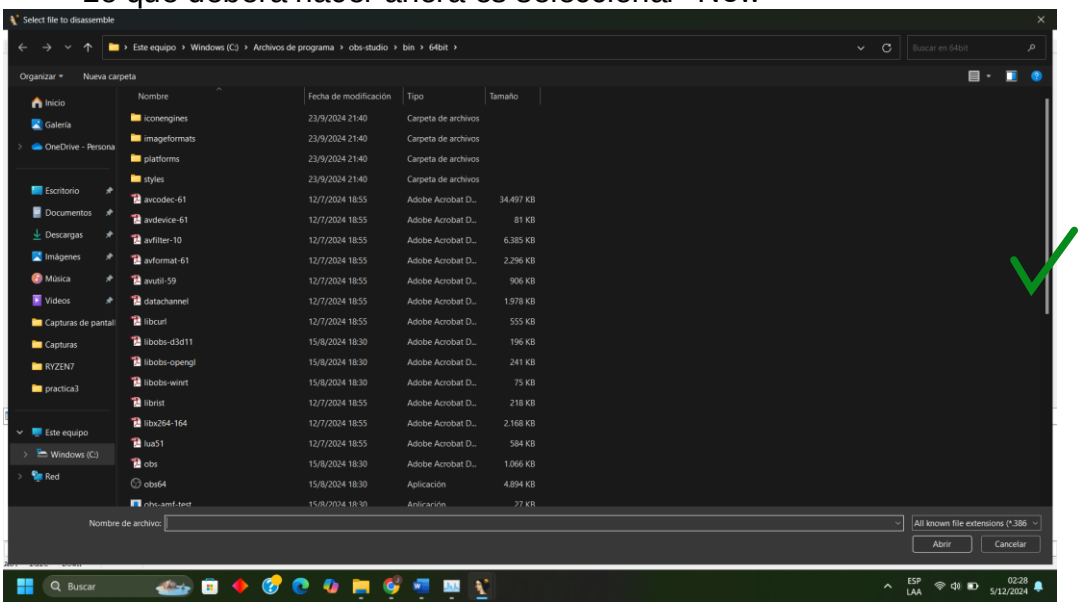
Una vez descargado e instalado deberán abrir el ejecutable .exe

Paso 3:

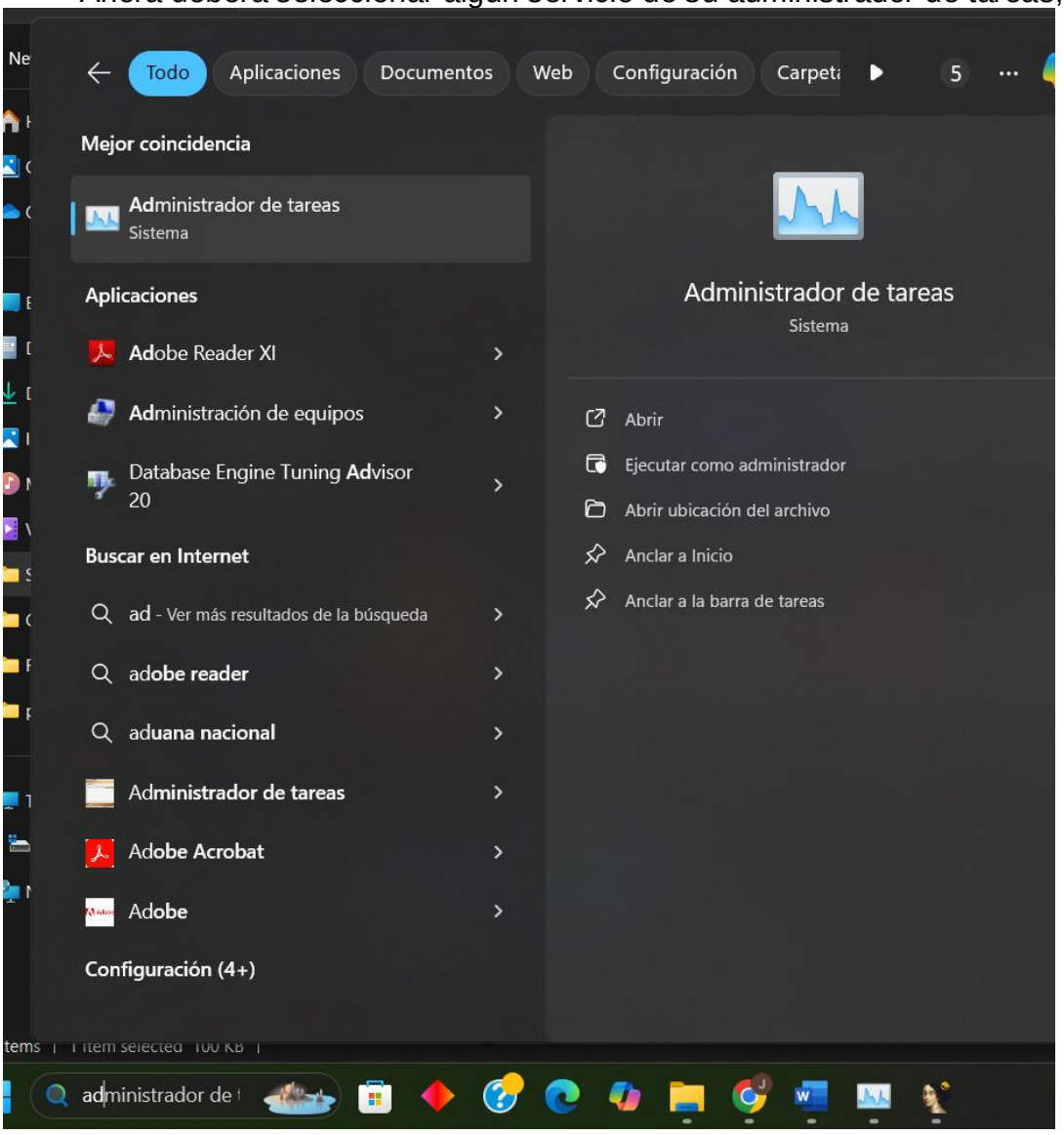


Procederemos a abrir un servicio en Windows

Lo que deberá hacer ahora es seleccionar “New”

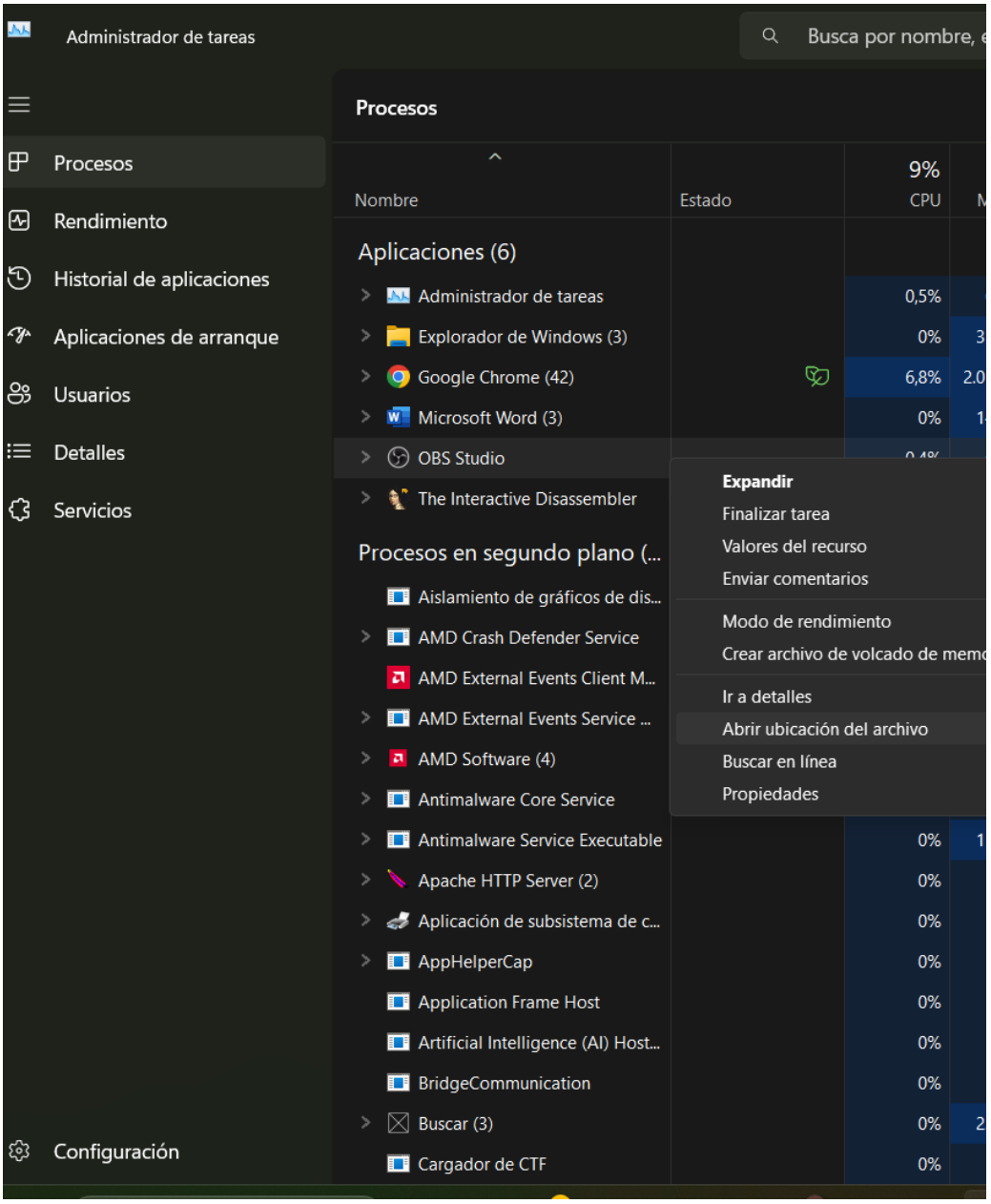


Ahora deberá seleccionar algún servicio de su administrador de tareas,

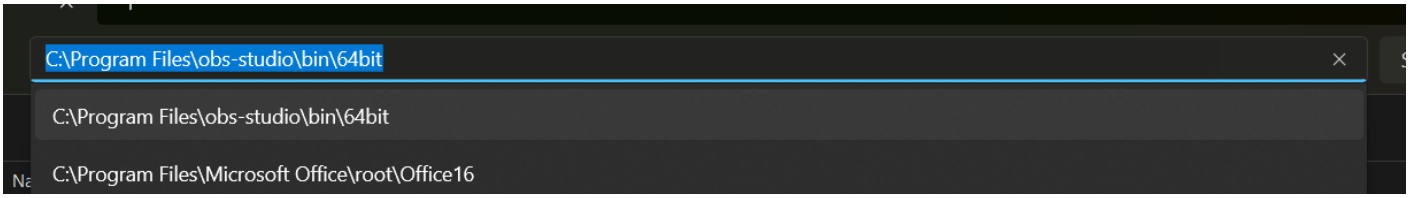


primeramente, vamos a abrir el administrador de tareas

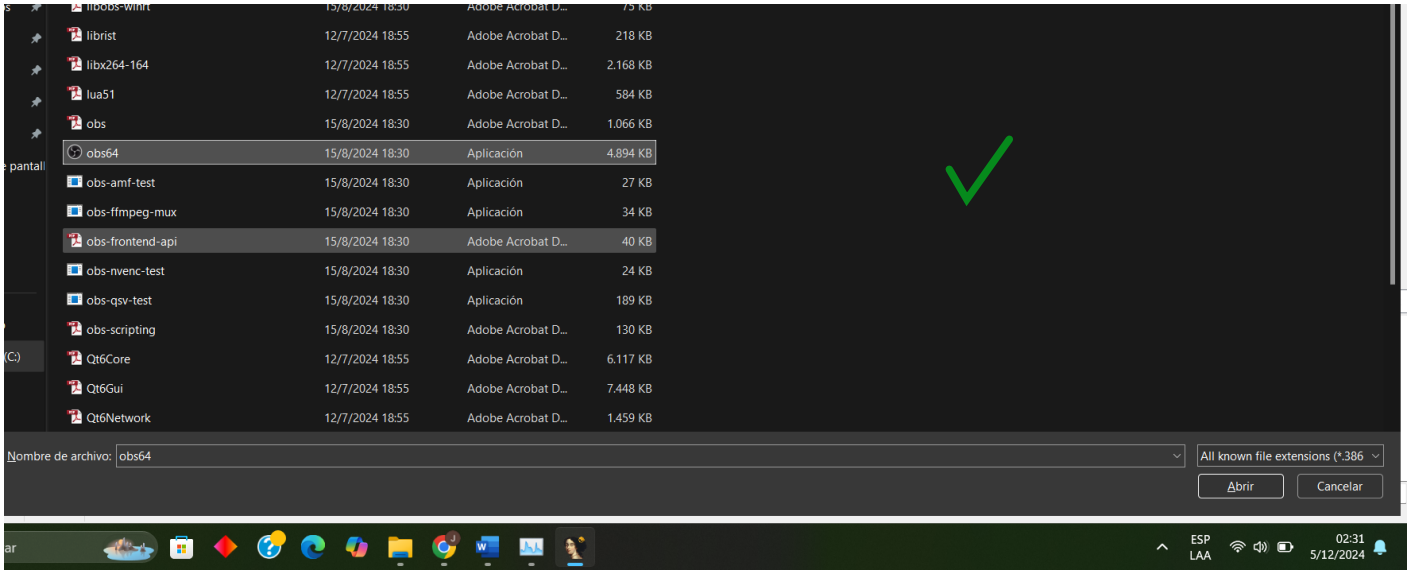
Ahora en la pestaña procesos deberá buscar cualquier servicio que se este ejecutando en tiempo real, y hacer un clic izquierdo sobre el servicio que le interesará ver el código ensamblador de este y después con un clic derecho seleccionar “Abrir ubicación del archivo”



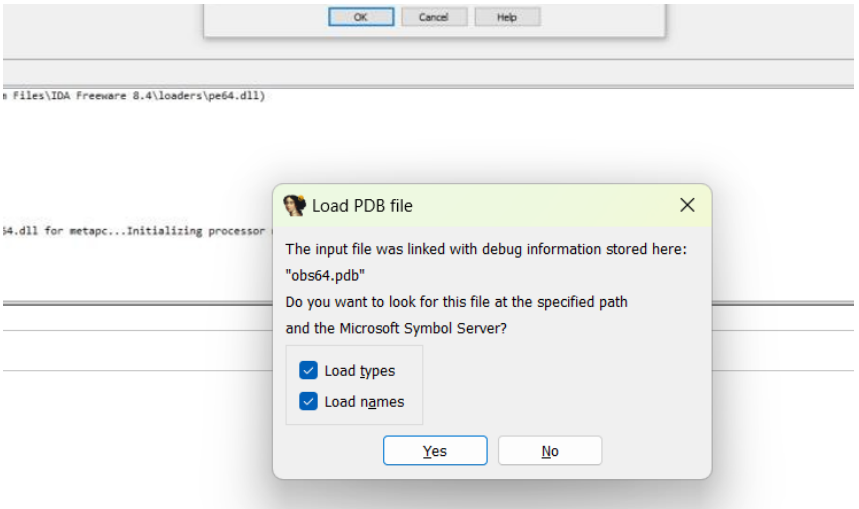
Ahora se deberá copiar la ruta en donde esta este servicio el cual es en este caso



Una vez copiada esta ruta se deberá colocar en la ventana donde IDA nos pidió que se debe añadir un servicio a analizar

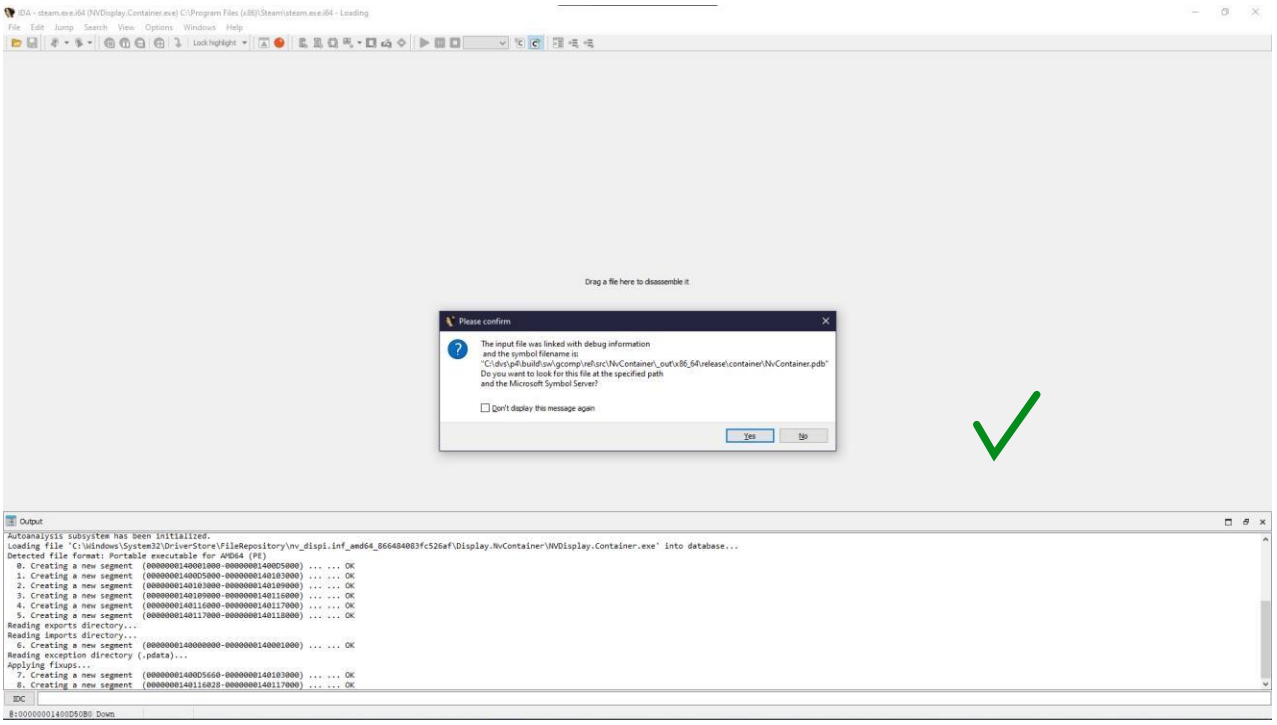


Una vez que coloquemos en guardar procederemos a desensamblar el servicio en este caso el “steam” tardará dependiendo el tamaño de



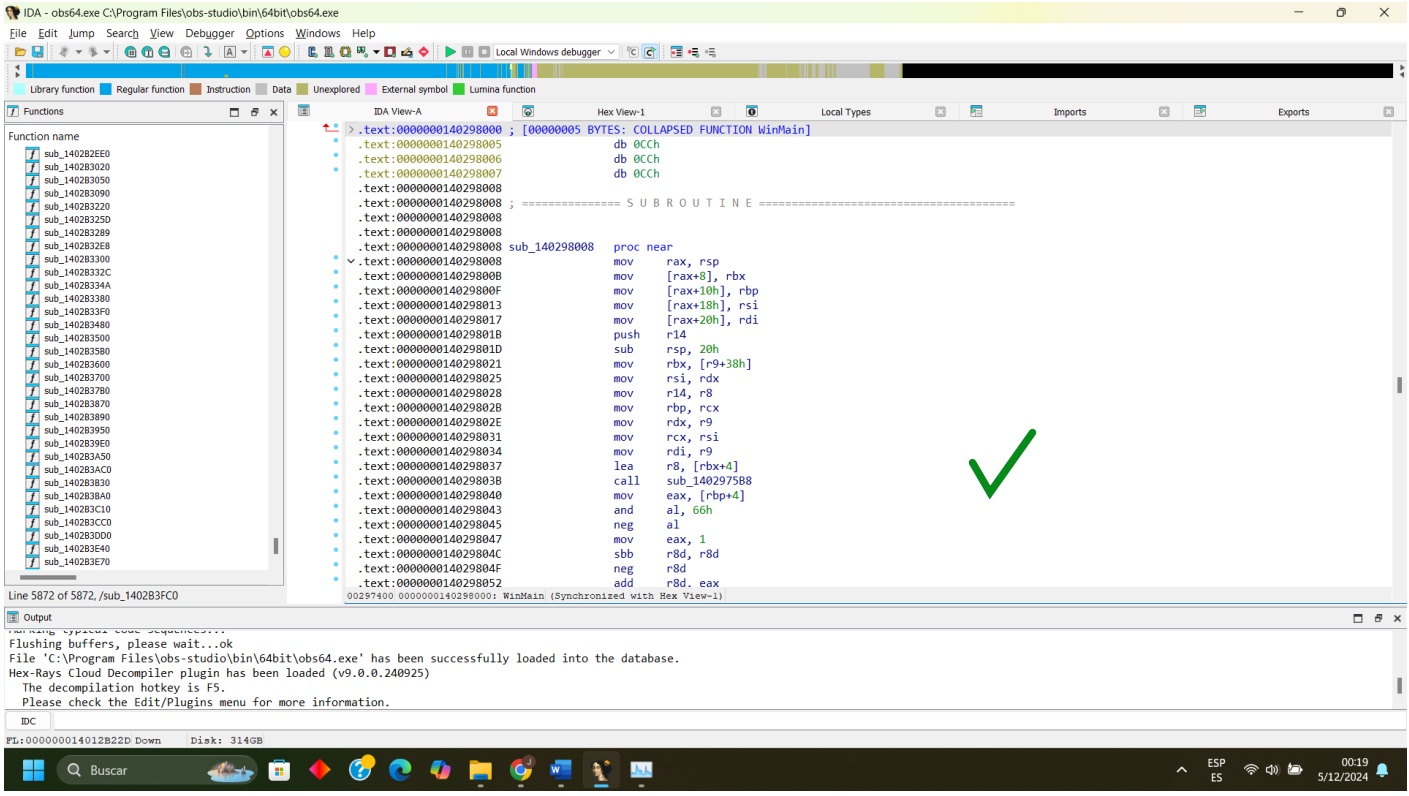
servicio a analizar. Dejaremos todo por definido y colocamos “ok”

Colocaremos “no”

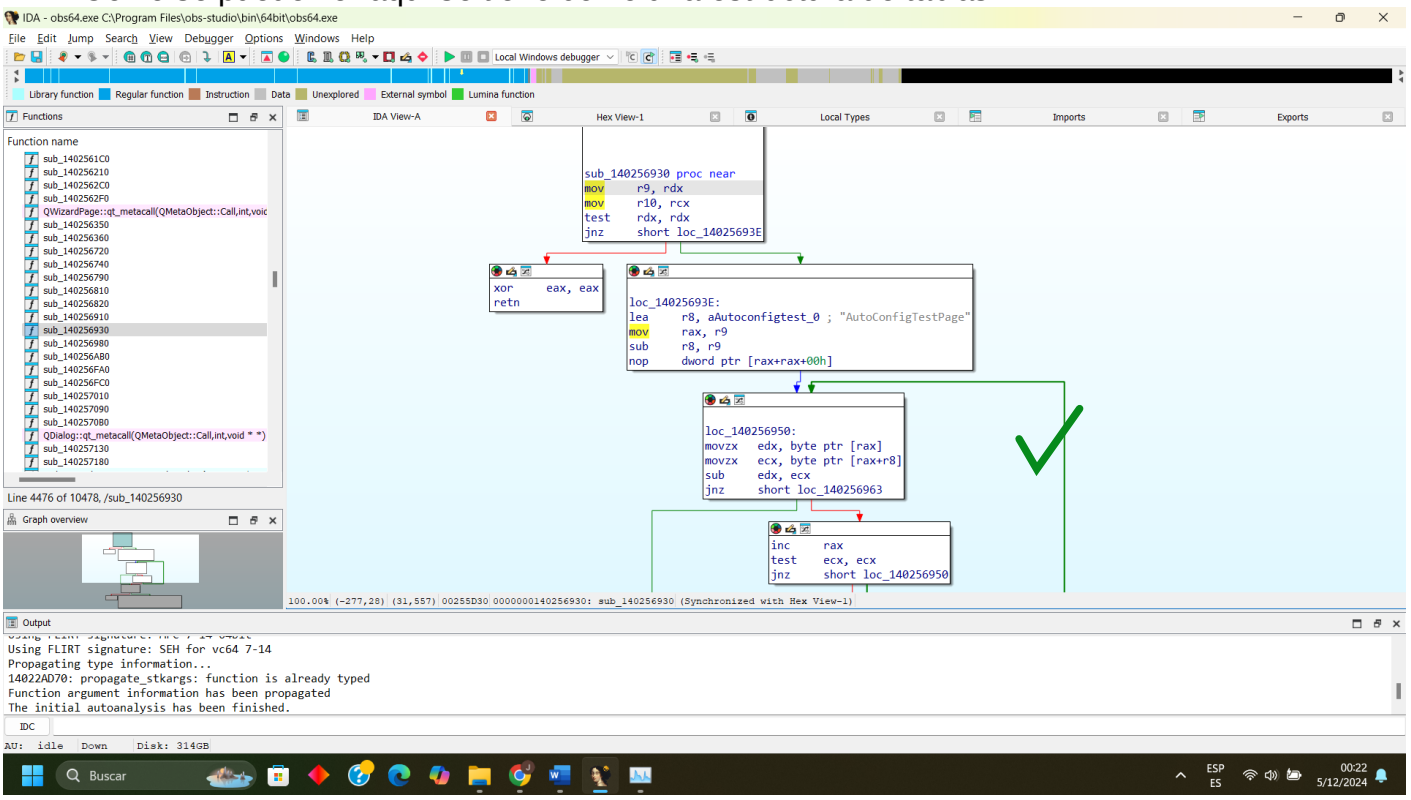
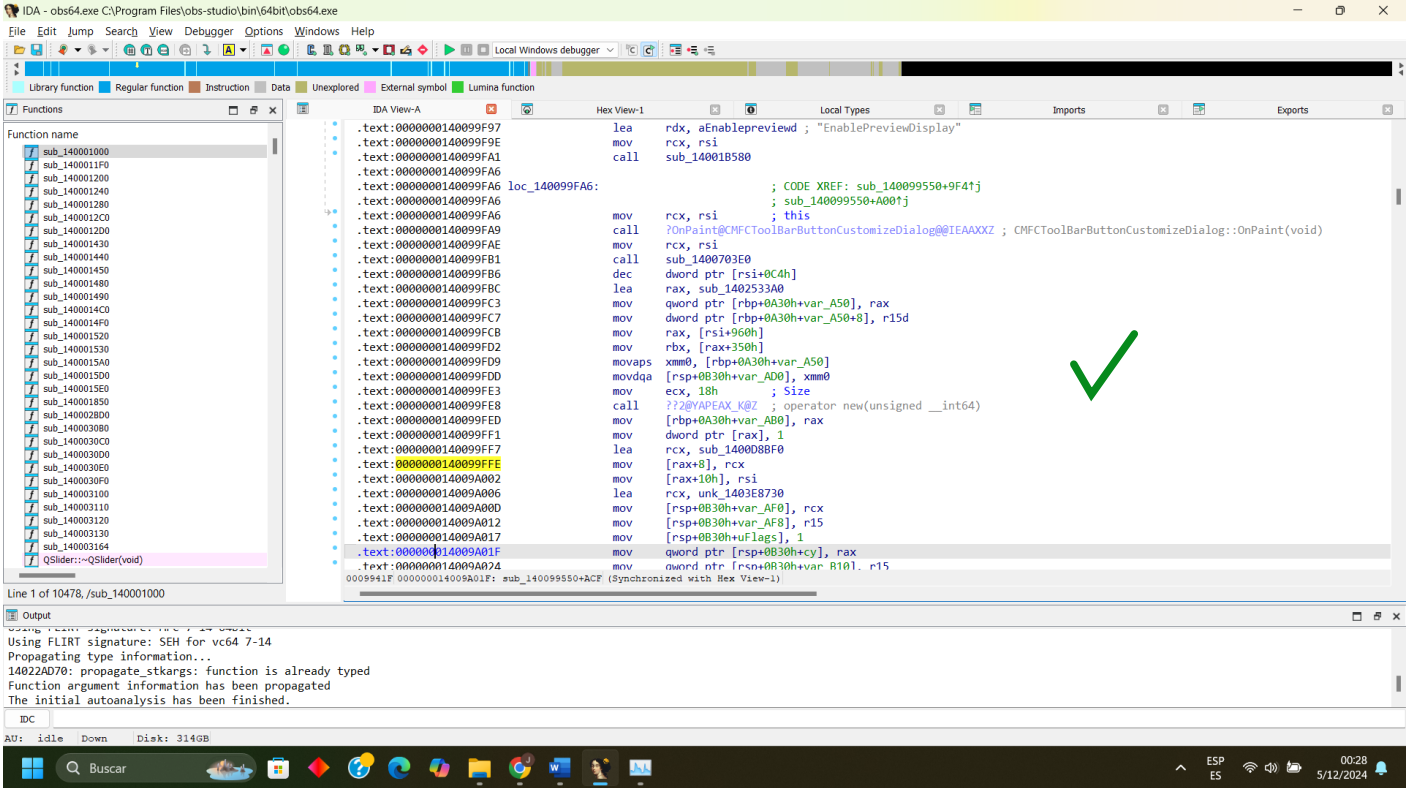


Paso 4:

Finalmente, se podrá ver código Assembler del servicio que hemos desensamblado



Como se puede ver aquí se tiene como una estructura de tablas



Aquí mismo se puede ver código Assembler