UNIVERSIDAD AUTÓNOMA "TOMAS FRÍAS" CARRERA DE INGENIERÍA DE SISTEMAS

Nombre: Pablo Mamani Oyola Arquitectura de computadoras (SIS-522) Materia: Ing. Gustavo A. Puita Choque Docente: Auxiliar: Univ. Aldrin Roger Perez Miranda 20/11/2024 Fecha publicación 05/12/2024 Fecha de entrega 1 Sede Potosí Grupo:



N° Práctica

C

Responda las siguientes preguntas de MANERA CONCISA

LAS RESPUESTAS DE MANERA DIGITAL en formato .pdf

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

R: El stack es un conteniendo de información acerca de la configuración inicial de la pila del programa y se utiliza con las siguientes instrucciones: Las instrucciones PUSH empujan valores a la pila mientras POP los saca, manteniendo un orden LIFO. Debe haber igual número de PUSH y POP para evitar errores de apilamiento, el CALL guarda la dirección del entorno del stack y salta a una subrutina y el RET recupera esa dirección para regresar al punto original.

2) Describe un escenario práctico donde el uso de ensamblador sería

más ventajoso que el uso de un lenguaje de alto nivel. (10 pts)

R: Un escenario práctico donde el ensamblador sería más ventajoso que el uso de lenguaje de alto nivel seria:

Donde un microcontrolador que controla un motor de precisión en un brazo robótico.

YA QUE AQUÍ EL ENSAMBLADOR ES IDEAL PORQUE:

Podría manejar sin interrupciones y con tiempos muy específicos (para generar señales PWM y para controlar los motores).

También podría ajustar directamente el ciclo de trabajo de una señal y así evitar retrasos.

También podría leer y escribir en puertos de entrada/salida para interactuar con sensores en tiempo real.

Y POR QUÉ NO USAR EN ESTE CASO UN LENGUAJE DE ALTO NIVEL:

Los lenguajes como C o Python, aunque son más fáciles de usar, suelen generar un código más grande y menos eficiente debido a la capa de abstracción que añaden.

Además, dependen de librerías y compiladores que pueden no ofrecer el nivel de control y personalización que se necesita en un hardware crítico.

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

R: En el MOV AX lo que pasa es que se mueve una carga que es el valor constante 5, por lo cual el valor de AX inicia con 5

En MOV BX lo que hace es que también se mueve una carga pero ahora el valor constante es 10, por lo cual BX inicia con 10

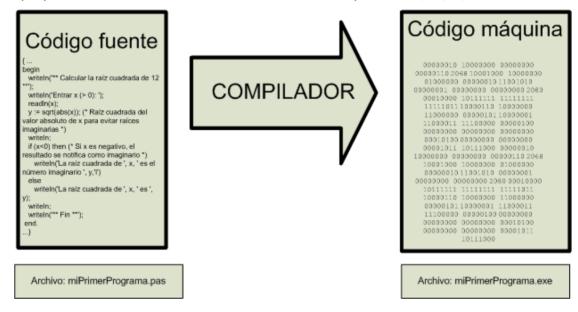
En la línea 3 lo que pasa es que al introducir ADD AX, BX suma el contenido de BX en el registro de

AX y el resultado se almacena en AX

Esto quiere decir que AX=AX+BX entonces: AX=5+10 entonces AX=15 por lo que ahora AX contiene 15 Y por último en MOV CX, AX lo que pasa es que se mueve el registro de AX en CX por lo tanto copia el valor de AX (15) y lo pone en el registro de CX.

En si lo que hace el código del lenguaje ensamblador es dar datos iniciales a AX y BX y luego efectuar una suma entre ellos y almacenar el resultado obtenido en AX en su registro y por último se mueve el registro ya sumado de AX en CX en su registro propio.

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



R: De esta manera funciona el compilador:

- 1. **Iniciación al análisis léxico:** el código se descompone en unidades mínimas de significado, denominadas tokens o lexemas.
- 2. **Progresión al análisis sintáctico:** se comprueba si los tokens se ajustan a las reglas sintácticas del lenguaje de programación, lo que lleva a la construcción de un árbol de análisis sintáctico.
- 3. **Transición al análisis semántico:** en esta fase se garantiza que los componentes utilizados en el código se ajustan a sus definiciones y son compatibles entre sí.
- 4. **Paso a la generación de código intermedio:** se produce la transformación del código fuente en una representación intermedia, más comprensible para la máquina.
- 5. **Avance hacia la optimización:** se produce la modificación del código intermedio para hacerlo más eficiente, sin alterar su funcionalidad original.
- **6.** Etapa final, la generación de código: se produce la conversión del código intermedio optimizado al lenguaje de destino, normalmente código máquina.

A lo largo de estas etapas, se observa el uso de tablas de símbolos para realizar un seguimiento de identificadores como nombres de variables, nombres de funciones, etc., junto con la notificación de cualquier problema encontrado durante la compilación mediante gestores de errores. Todo este proceso garantiza la transformación del código fuente de alto nivel en código máquina de bajo nivel que pueda ejecutar un ordenador.

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

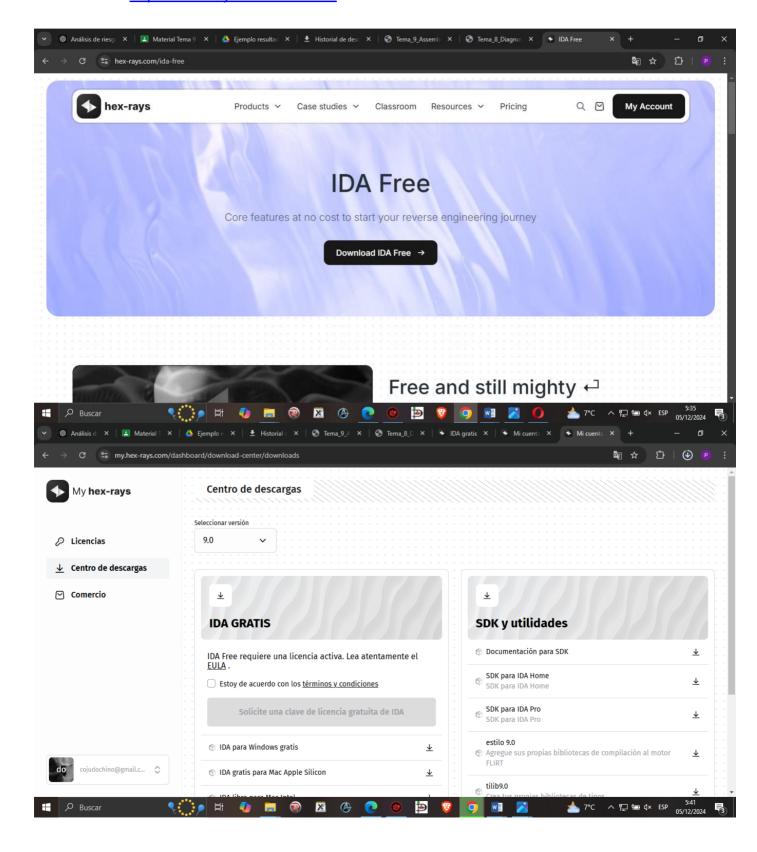
IDA: Es una de las herramientas más conocidas y potentes para el análisis de código binario y desensamblado. En este laboratorio se

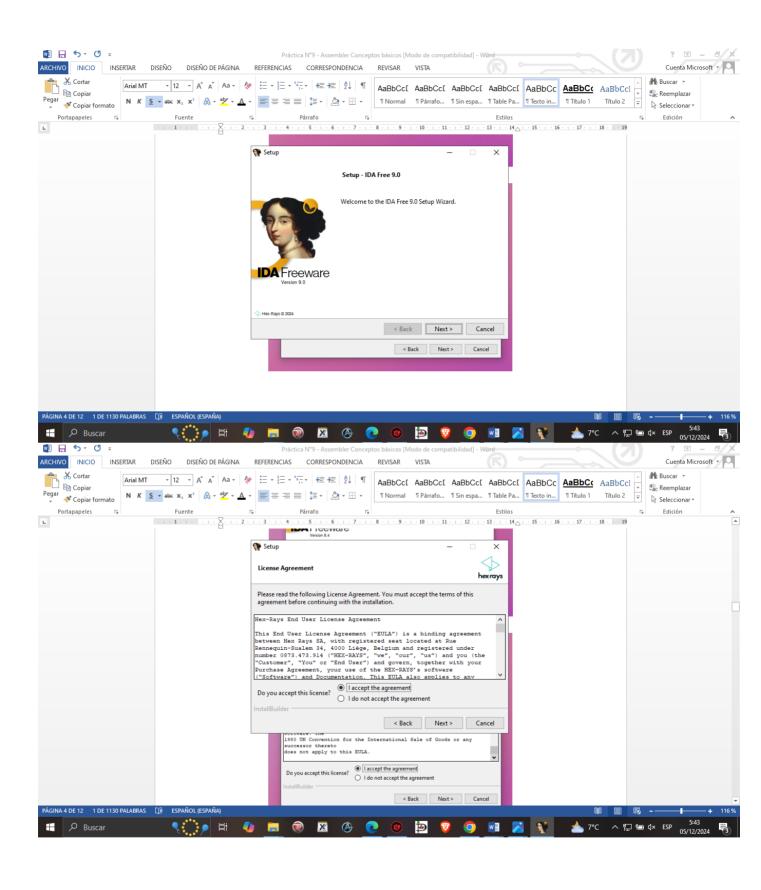
instalará IDA FREE pero también se tiene la versión de paga IDA PRO

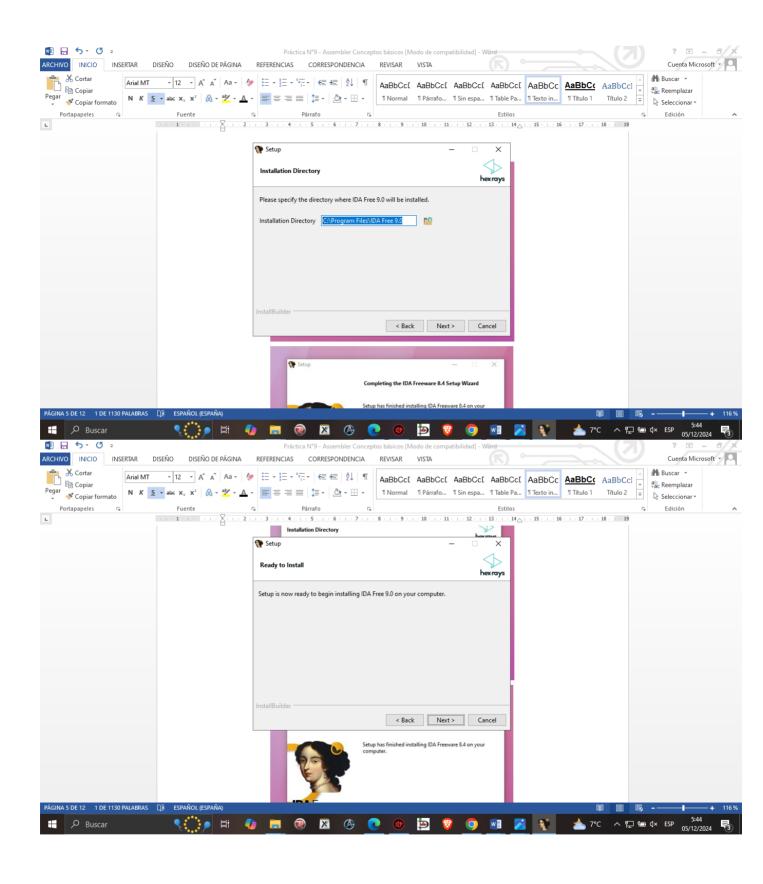
Paso 1:

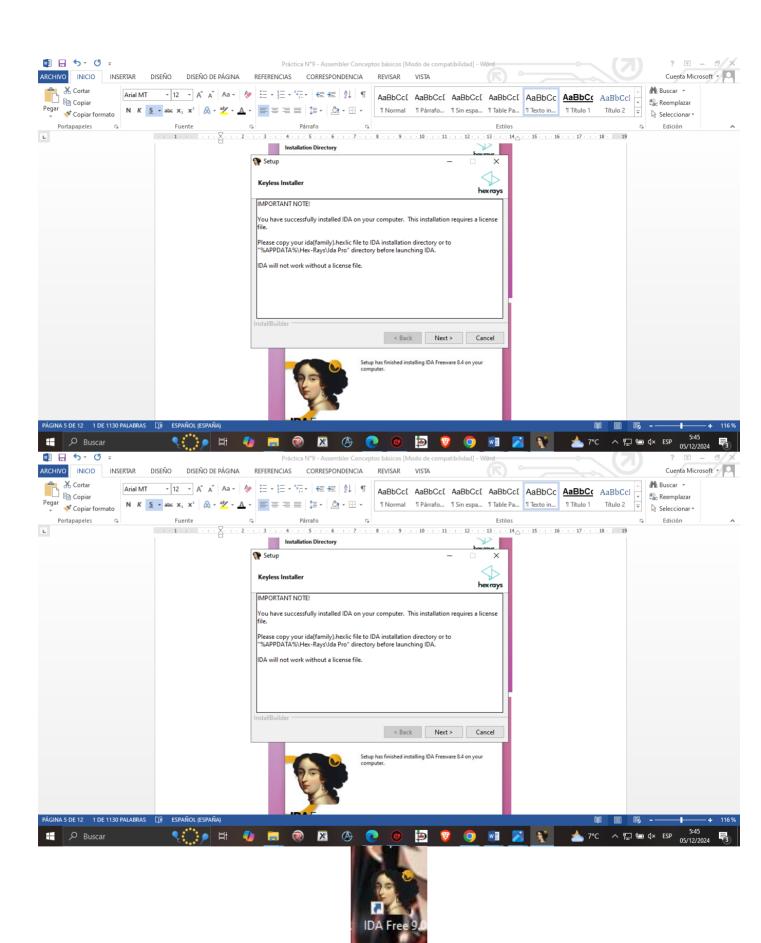
Descargar el software IDA FREE el cual lo podrá a hacer del siguiente

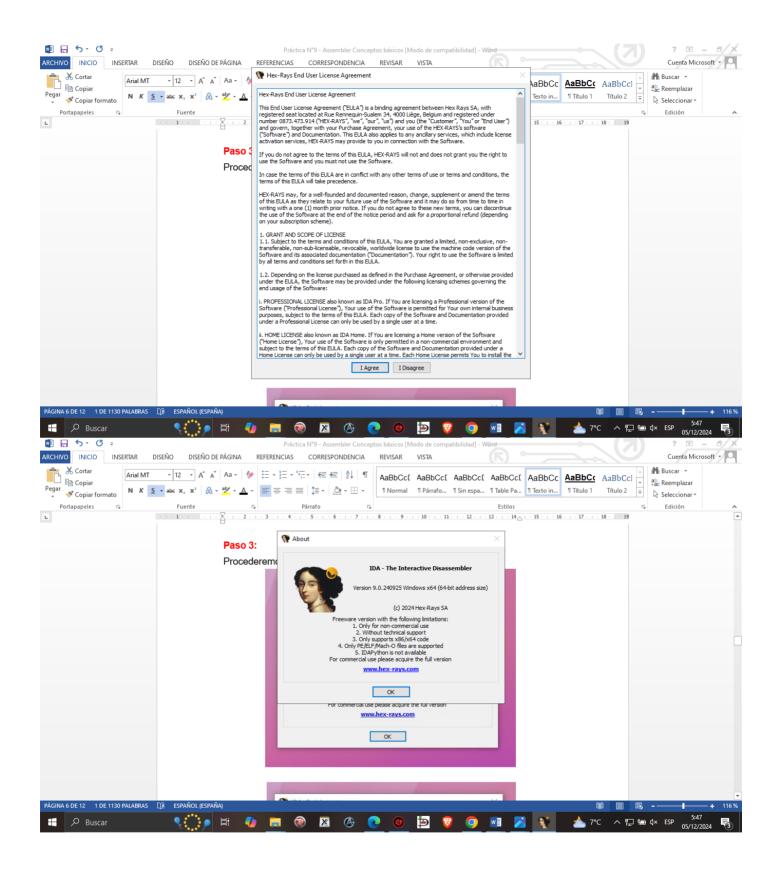
enlace: https://hex-rays.com/ida-free/

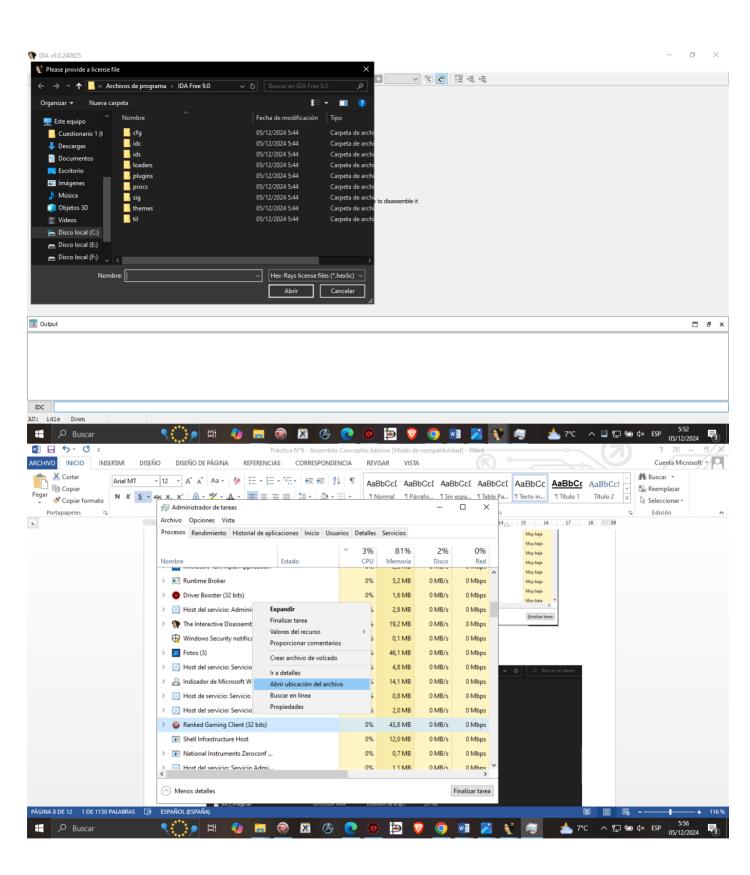


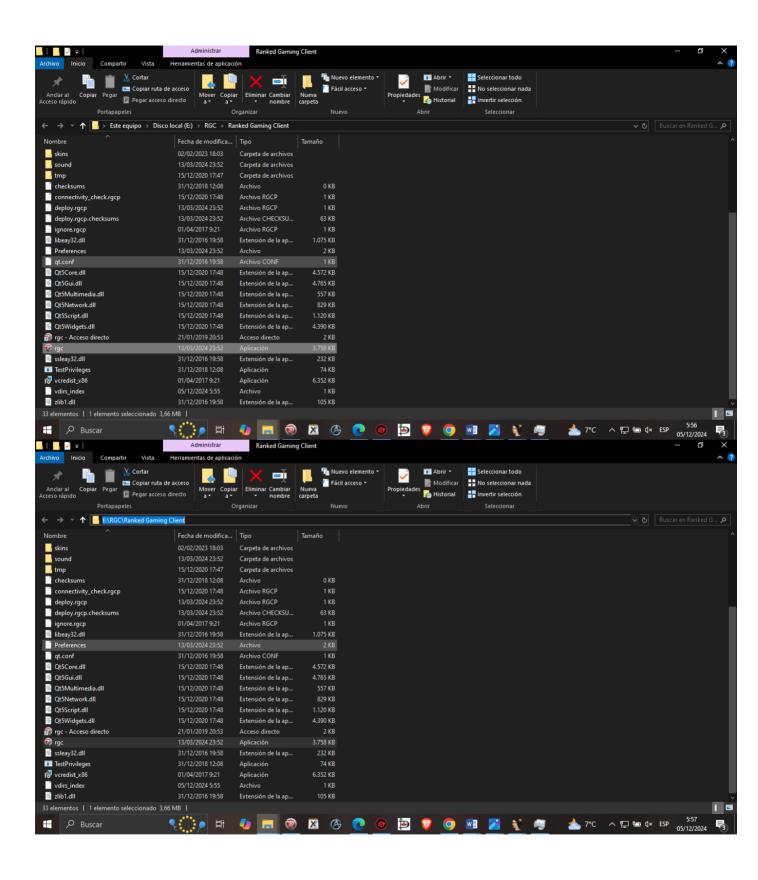


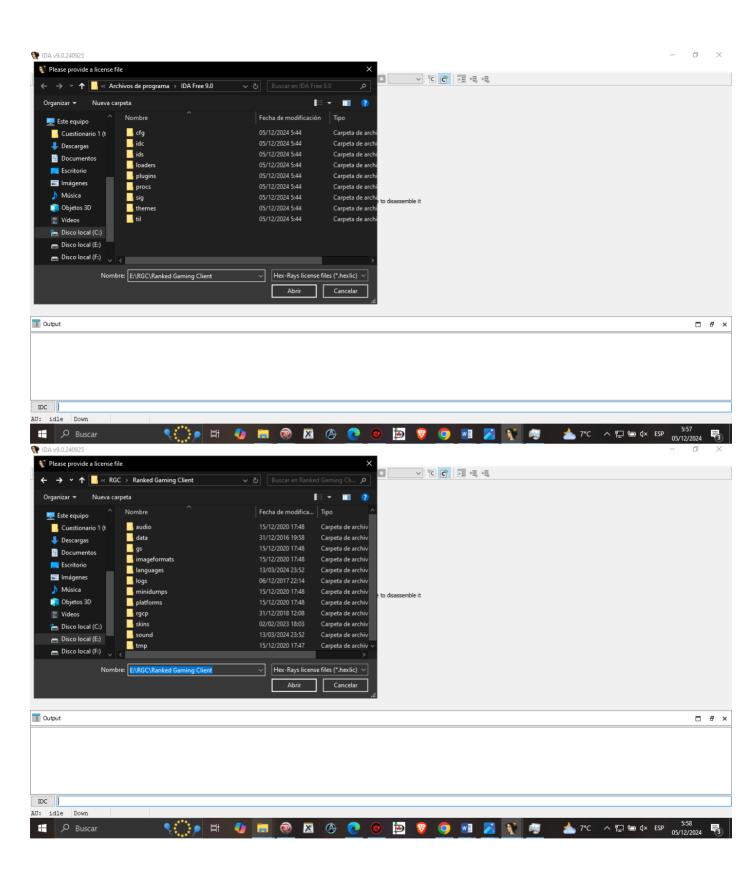


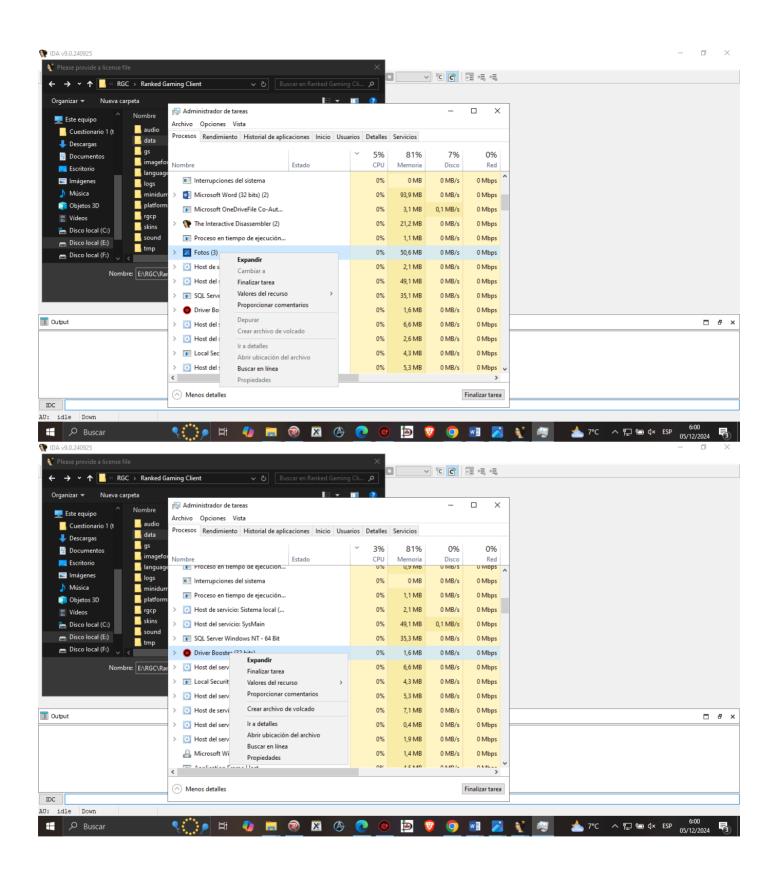


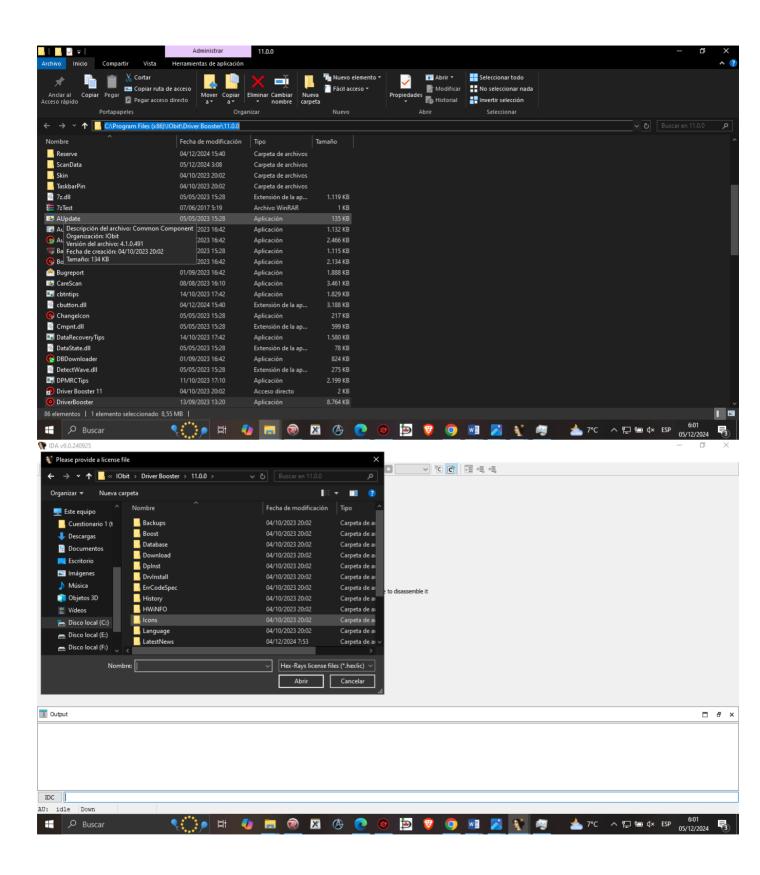


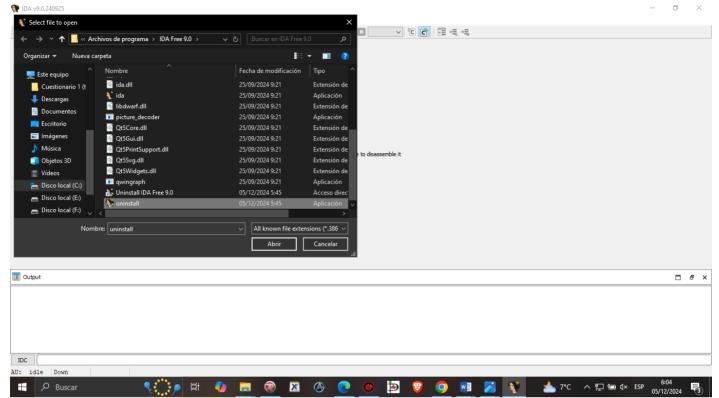












LOS SIGUIENTES PASOS NO PUDE REALIZAR YA QUE ME PEDIA LICENCIA Y NO ME DEJABA ABRIR ARCHIVOS COMO EN EL EJEMPLO QUE NOS DIA AUXILIAR Y NO SUPE COMO CONSEGUIR LA LICENCIA QUE PEDIA HASTA AQUÍ PUDE LLEGAR.