


<u>UNIVERSIDAD AUTÓNOMA “TOMAS FRÍAS”</u> <u>CARRERA DE INGENIERÍA DE SISTEMAS</u>				
Materia:	Arquitectura de computadoras (SIS-522)			N° Práctica
Docente:	Ing. Gustavo A. Puita Choque			
Auxiliar:	Univ. Aldrin Roger Perez Miranda			9
16/06/2024	Fecha publicación			
01/07/2024	Fecha de entrega			
Grupo:	1	Sede	Potosí	

ESTUDIANTE: Univ. José Felipe Mamani Azurduy

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

R= En el lenguaje ensamblador el 'stack' es una estructura de datos que sigue el fundamental y único principio del LIFO (Last In, First Out) que en español sería Último en entrar, Primero en salir. Este stack se utiliza primordialmente para almacenar datos temporales como los registros o las direcciones de retorno de las subrutinas. Las operaciones básicas son PUSH (para añadir datos) y POP (para quitar datos). Es fundamental para la gestión eficiente de la memoria durante la ejecución del programa.

2) Describe un escenario práctico donde el uso de ensamblador sería más ventajoso que el uso de un lenguaje de alto nivel.

R= Un escenario donde el uso del ensamblador es más ventajoso que el uso de un lenguaje de alto nivel, es en el desarrollo de controladores de dispositivos o sistemas embebidos. Permitiendo las siguientes cosas:

- Acceso directo a interactuar con registros específicos de los dispositivos.
- Control totalitario para poder maximizar la velocidad y minimizar la latencia.
- Utilizar eficientemente la memoria con un código más compacto y eficiente que los lenguajes de alto nivel, muy recomendado cuando el rendimiento y el control sobre nuestro hardware son críticos.

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

R= Línea 1: MOV AX, 5

- Instrucción: MOV
- Descripción: Mueve el valor 5 al registro AX.
- Qué hace: Asigna el valor 5 al registro AX. Después de esta instrucción, el registro AX contiene 5.

Línea 2: MOV BX, 10

- Instrucción: MOV
- Descripción: Mueve el valor 10 al registro BX.

- Qué hace: Asigna el valor 10 al registro BX. Después de esta instrucción, el registro BX contiene 10.

Línea 3: ADD AX, BX

- Instrucción: ADD
- Descripción: Suma el valor del registro BX al registro AX y almacena el resultado en AX.
- Qué hace: Suma el contenido de BX (que es 10) al contenido de AX (que es 5). Después de esta instrucción, AX contiene 15 (5 + 10).

Línea 4: MOV CX, AX

- Instrucción: MOV
- Descripción: Mueve el valor del registro AX al registro CX.
- Qué hace: Copia el valor contenido en AX (que es 15 después de la suma) al registro CX. Después de esta instrucción, CX contiene 15.

4) Explique detalladamente cómo funcionan los compiladores

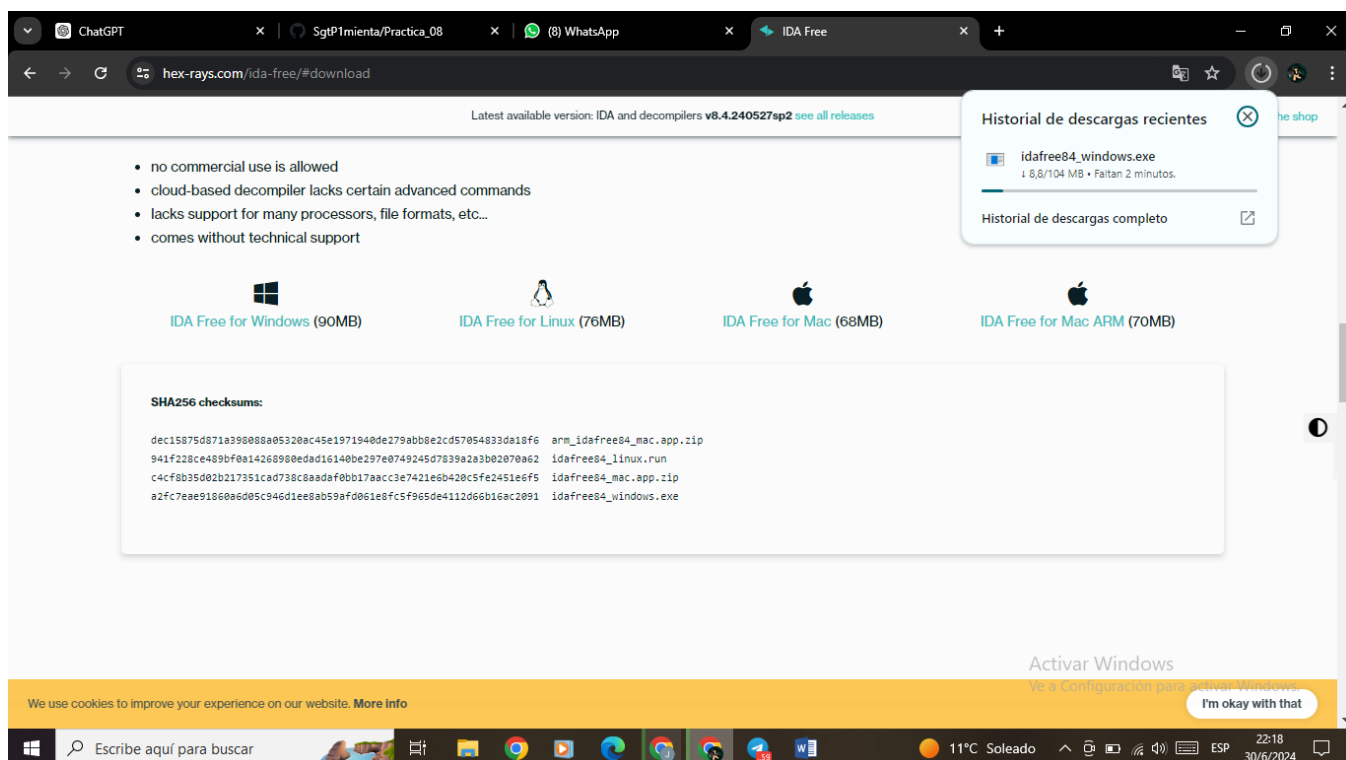
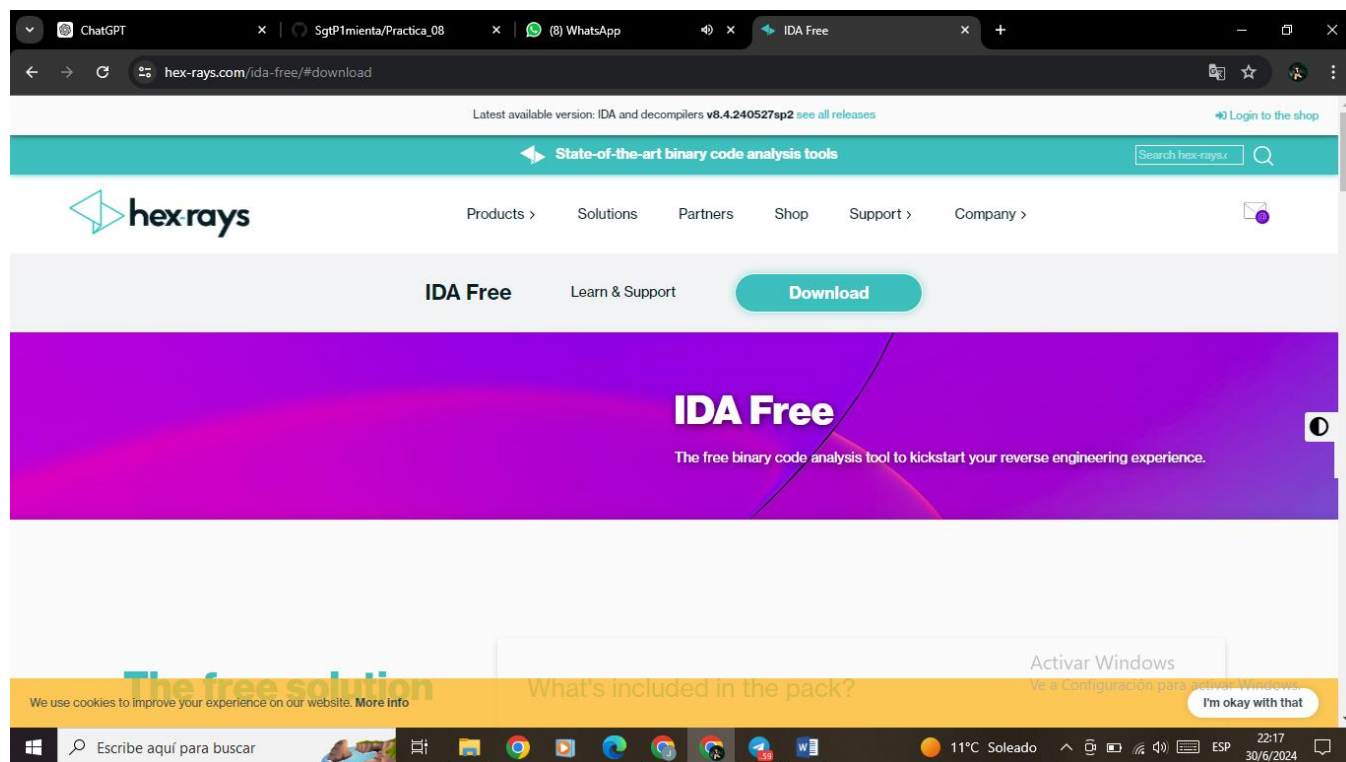
R= Para explicar a detalle cómo funcionan los compiladores, podemos comenzar conceptualizando que estos, son programas en síntesis, que traducen el código fuente escrito en un lenguaje de programación de alto nivel (como C, C++, Java u otro) a un lenguaje de bajo nivel o código máquina (binario) que puede ser ejecutado directamente por el hardware de nuestro ordenador. Este proceso se lleva a cabo en muchas etapas de las cuales albergaremos:

- **Análisis Léxico (Lexical Analysis):** Divide el código en tokens que son unidades básicas como palabras clave, identificadoras u operadores, y luego un analizador se encarga de escanear el código y engrupar caracteres en esos tokens.
- **Análisis Sintáctico (Syntax Analysis):** Hará la verificación de la estructura del código y creará un árbol de sintaxis abstracta o en sus siglas AST, así organizando los tokens en una estructura jerárquica.
- **Análisis Semántico (Semantic Analysis):** Corroborar que el código tenga sentido semántico que en otras palabras serían los tipos de datos y la existencia de variables siguiendo con la correspondencia de tipos y la correcta utilización de operadores.
- **Generación de Código Intermedio (Intermediate Code Generation):** Traduce el AST a un código intermedio independiente de la máquina haciendo esto en un formato más fácil de ser optimizado.
- **Optimización del Código (Code Optimization):** Incrementa la eficiencia del código intermedio, o sea, eliminará el código innecesario así decrementando el tiempo de ejecución y el uso de memoria.
- **Generación de Código (Code Generation):** Traducirá el código intermedio a instrucciones de máquina excepcionalmente o sea que ya tendríamos las instrucciones de ensamblador.
- **Enlazado (Linking):** Junta varios módulos de código objeto en un ejecutable.

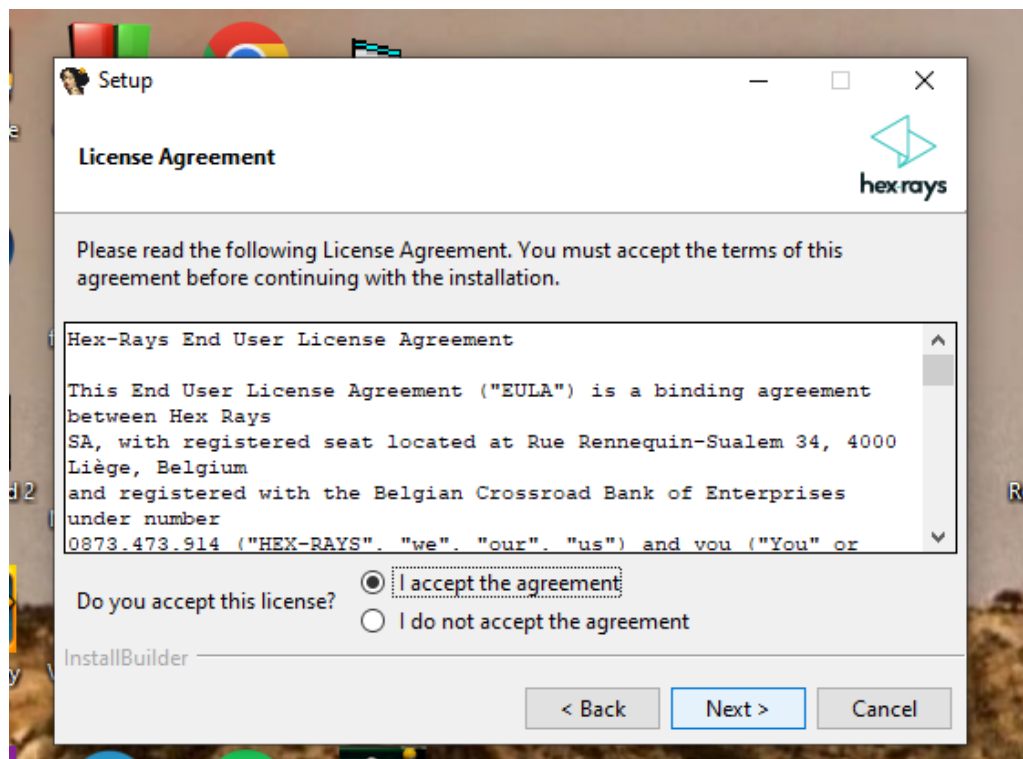
5) Realizar capturas de pantalla del siguiente procedimiento:

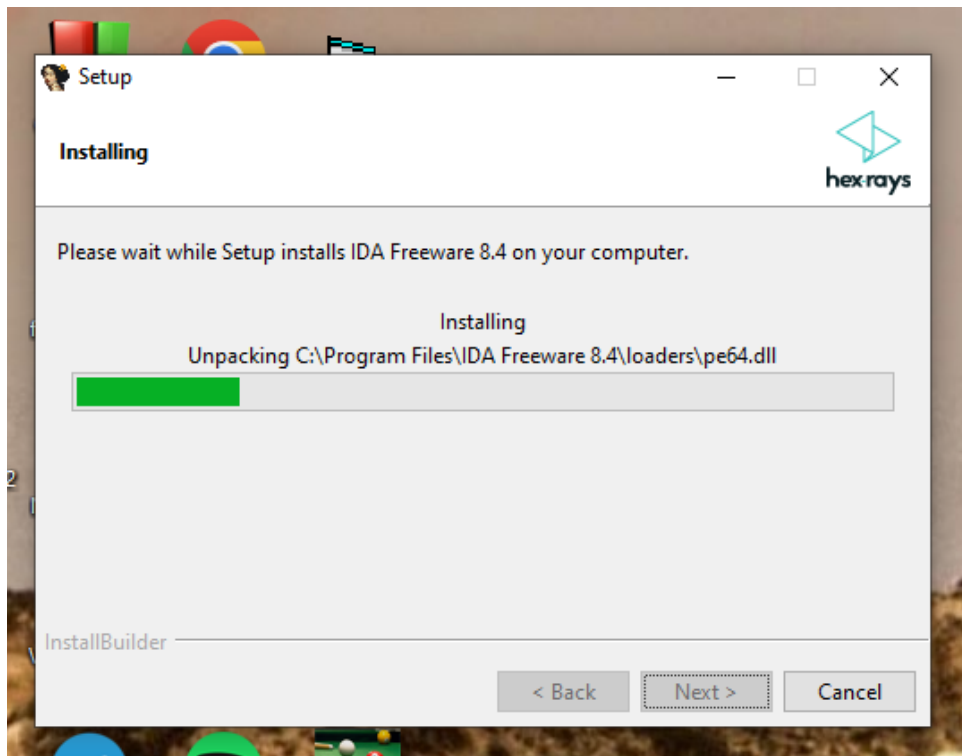
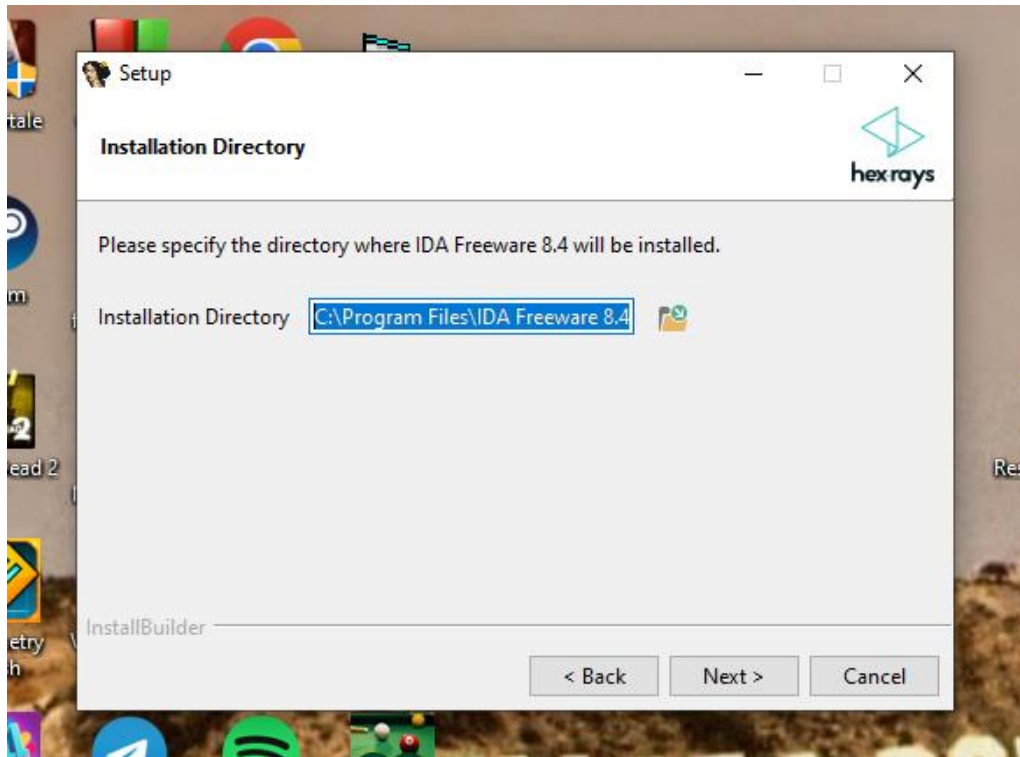
EL PROCEDIMIENTO LO DEBE HACER COMO UN LABORATORIO PASO A PASO Y EXPLICAR QUE ES LO QUE SE ESTA HACIENDO CON SU RESPECTIVA CAPTURA USTED DEBE SELECCIONAR CUALQUIER SERVICIO DE SU PREFERENCIA

PASO 1:



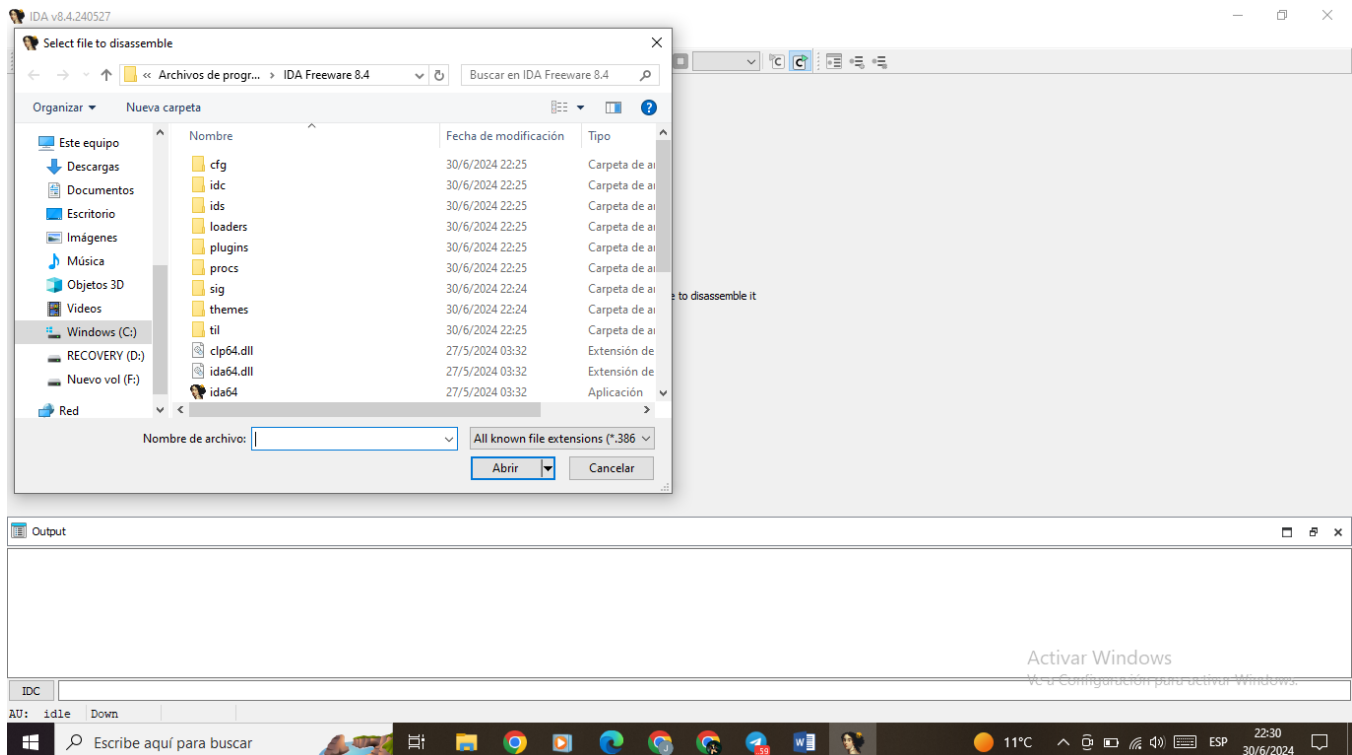
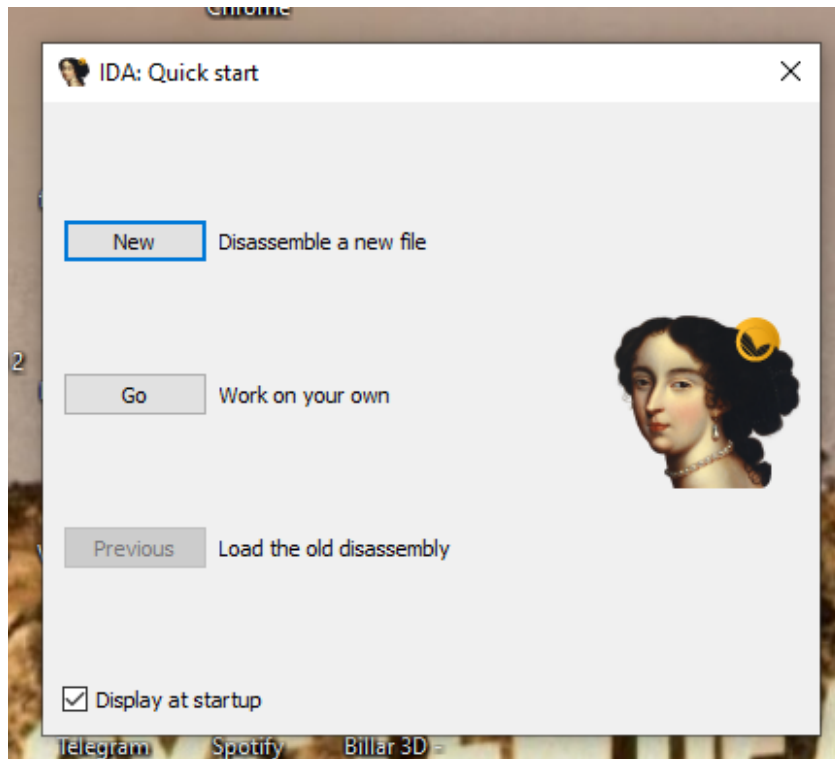
PASO 2:

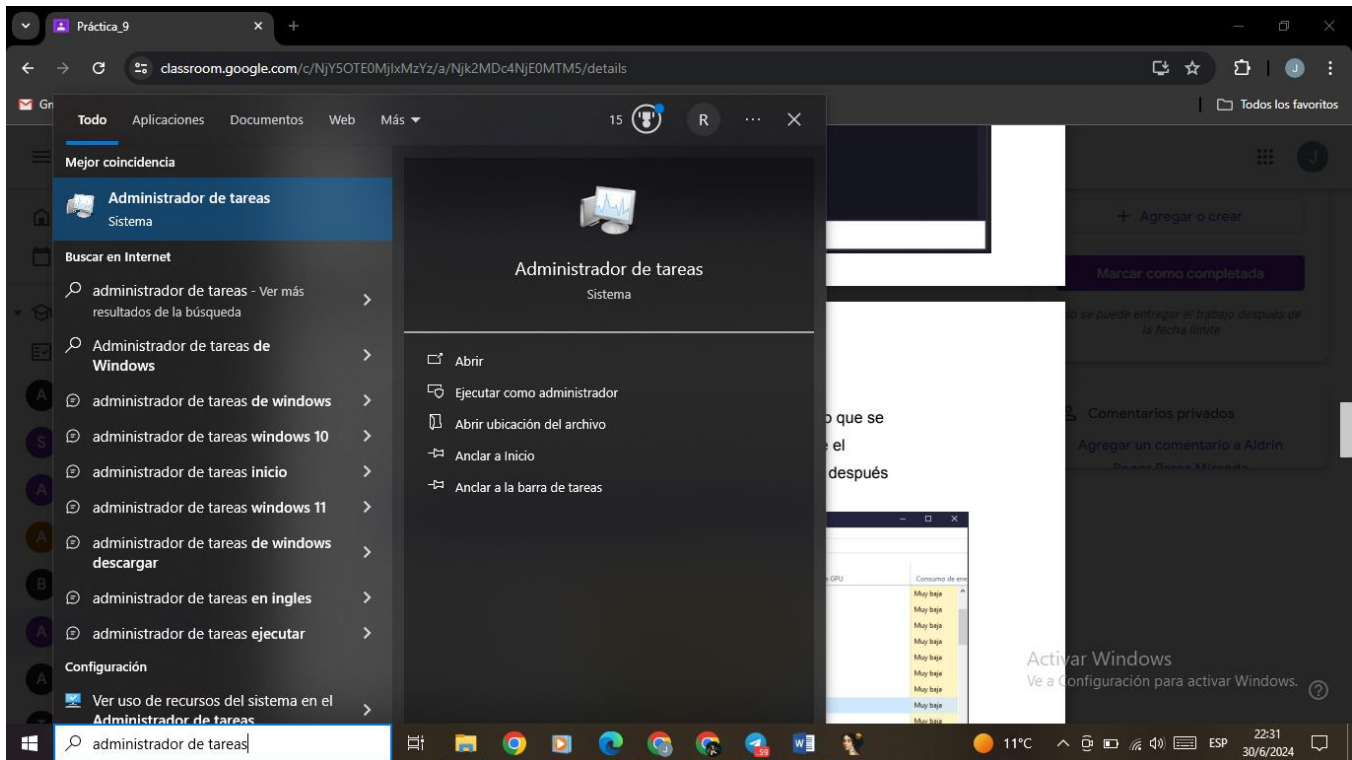






PASO 3:





Administrador de tareas

Archivo Opciones Vista

Procesos Rendimiento Historial de aplicaciones Inicio Usuarios Detalles Servicios

Nombre

Estado

4% CPU

81% Memoria

3% Disco

0% Red

0% GPU

Motor de la GPU

Consumo de ...

Tendencia de ...

Aplicaciones (6)

Administrador de tareas

Explorador de Windows

Google Chrome (7)

Microsoft Word (32 bits) (2)

Telegram Desktop (32 bits) (2)

The Interactive Disassembler

Procesos en segundo plano

64-bit Synaptics Pointing Device

AggregatorHost

Antimalware Core Service

Antimalware Service

Aplicación de subprocesos

Application Frame Host

Bonjour Service

BraveSoftware Update

BraveSoftware Update (32 bits)

Búsqueda

Cargador de CTF

Expandir

Finalizar tarea

Valores del recurso

Enviar comentarios

Crear archivo de volcado

Ir a detalles

Abrir ubicación del archivo

Buscar en línea

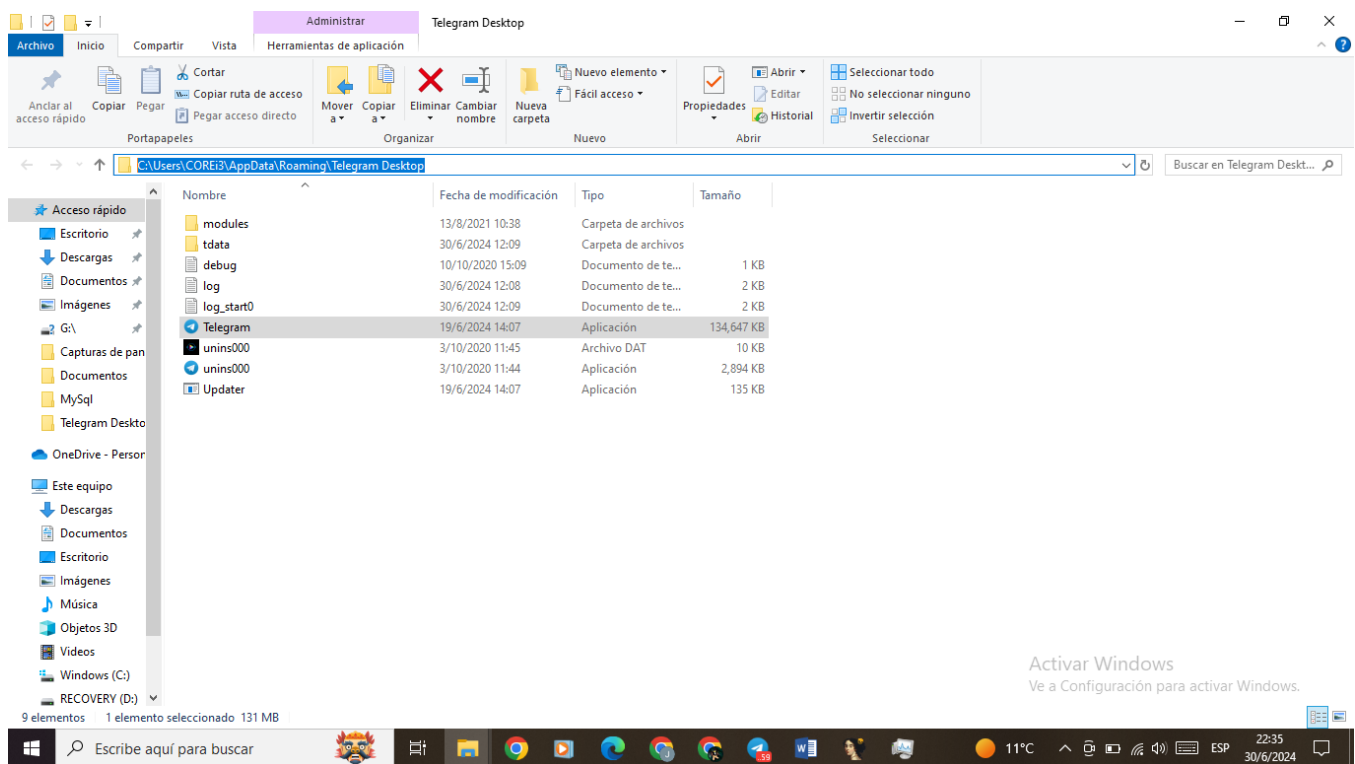
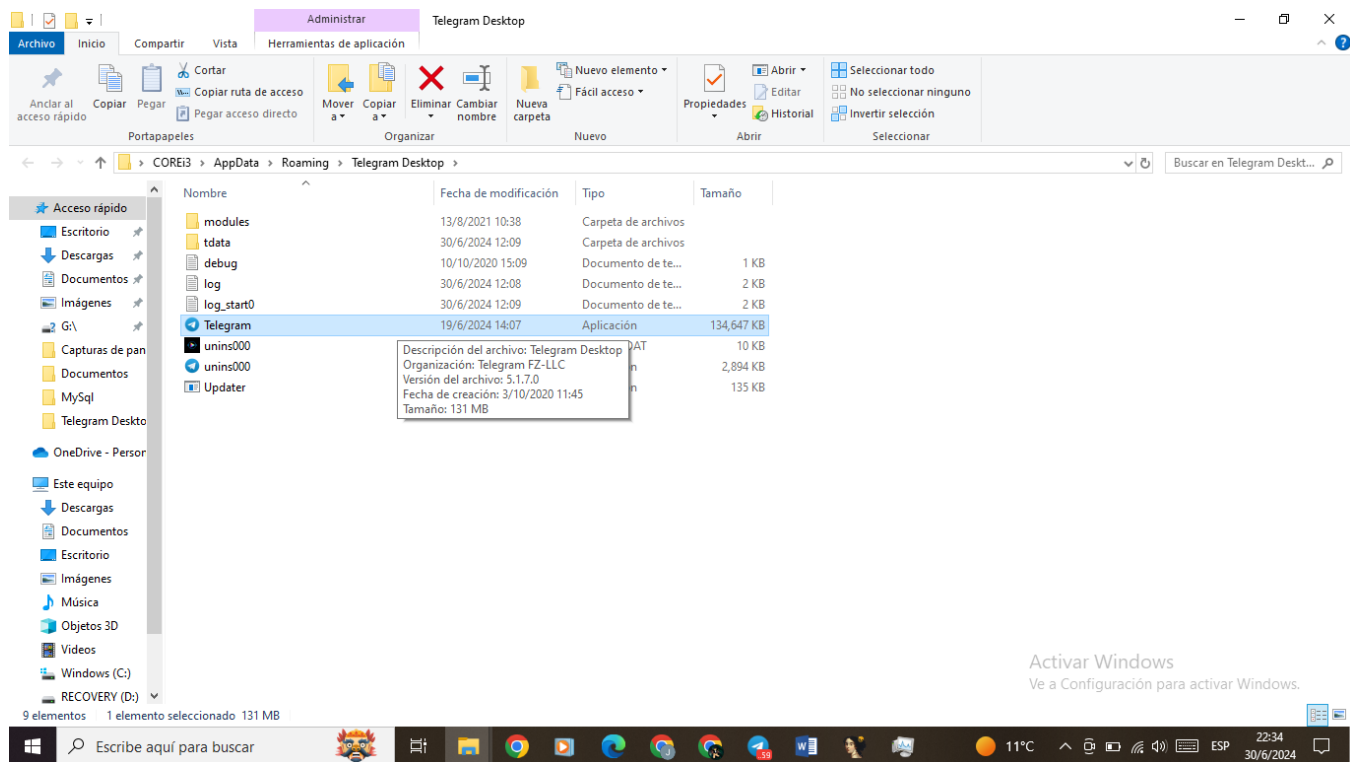
Propiedades

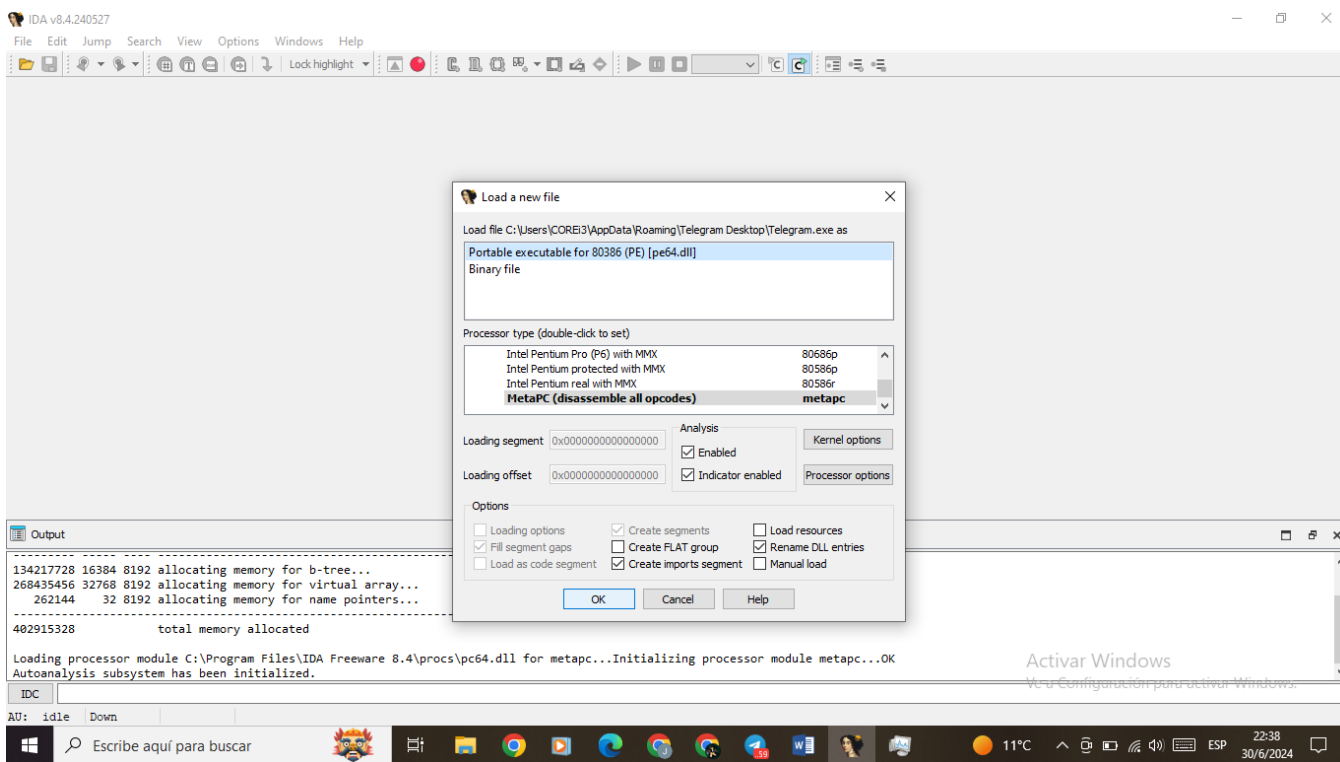
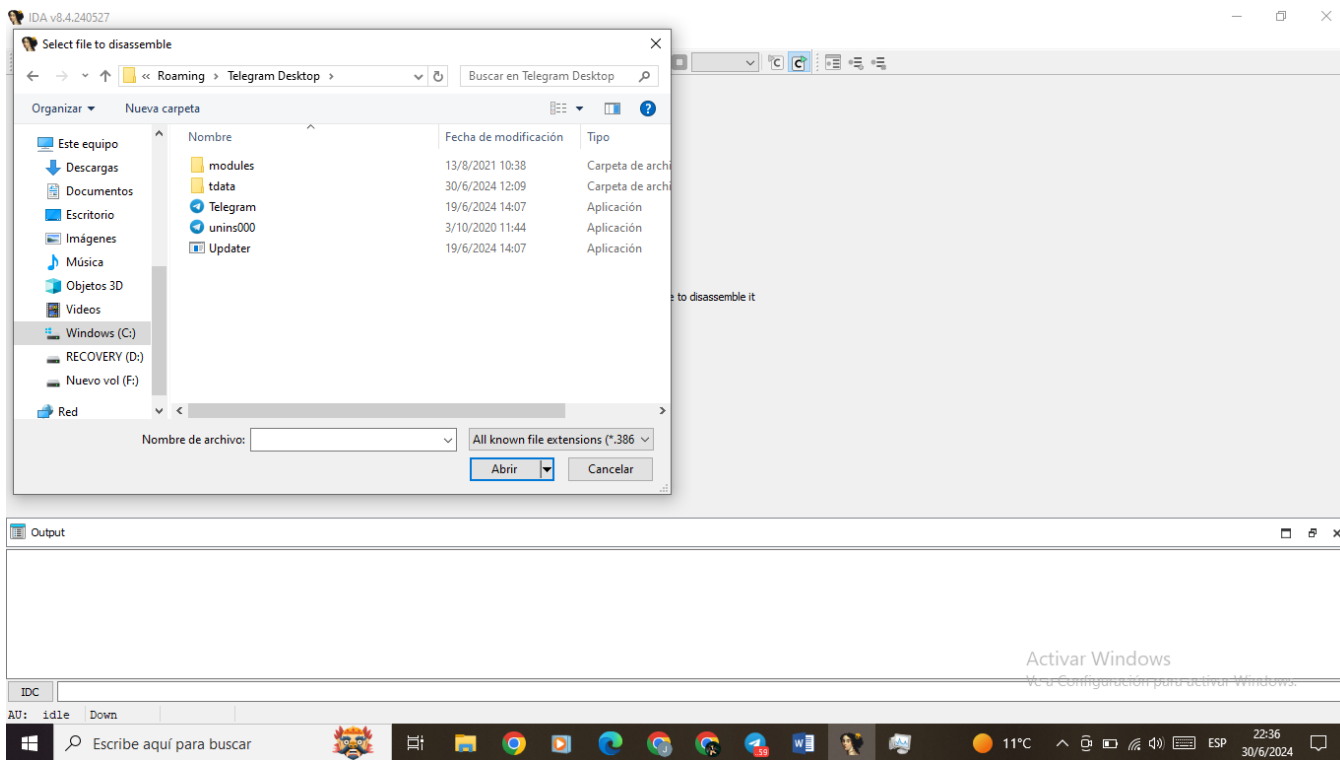
Menos detalles

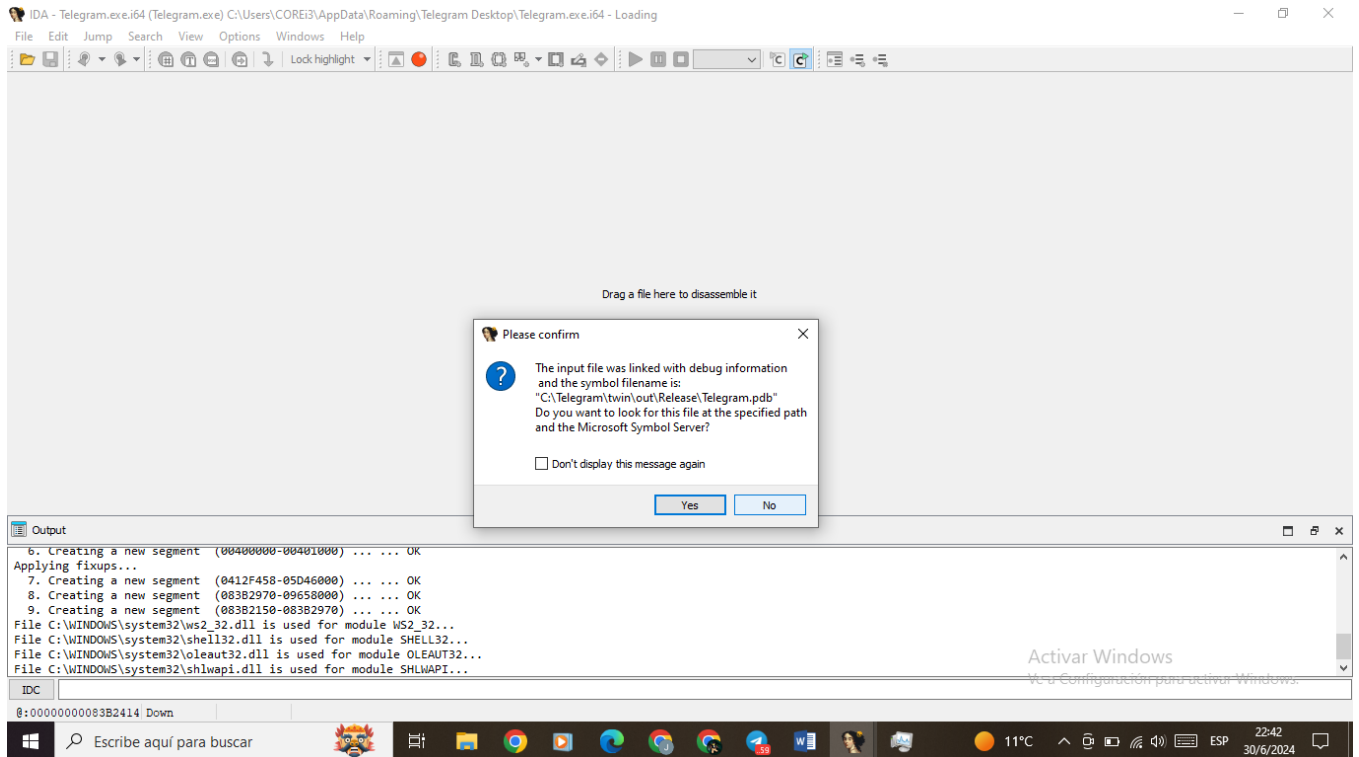
Finalizar tarea

Finalizar Windows

Ve a Configuración para activar Windows.







PASO 4:

