	Nombre: Cristian David Paco Bravo
	Carrera: Ingeniería de Sistemas
	Materia: Arquitectura de computadoras
	Docente: Ing. Gustavo A. Puita Choque
Fecha de entrega: 05-12-24	Auxiliar: Univ. Aldrin Roger Perez Miranda

PRACTICA # 9

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

El **stack** (pila) es una estructura de datos utilizada para almacenar información de manera temporal en la memoria durante la ejecución de un programa. Funciona bajo el principio LIFO (Last In, First Out), es decir, el último elemento en entrar es el primero en salir. En el contexto del ensamblador, el stack se utiliza para almacenar direcciones de retorno, variables locales y parámetros de función.

El stack en ensamblador se usa de la siguiente forma:

- **PUSH**: Instrucción que se utiliza para colocar un valor en el tope de la pila.
- **POP**: Instrucción que se utiliza para retirar el valor en el tope de la pila.
- **CALL**: Instrucción que guarda la dirección de retorno en la pila y salta a una subrutina.
- **RET**: Instrucción que recupera la dirección de retorno desde la pila y vuelve a la ejecución desde esa dirección.

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

Un escenario práctico donde el ensamblador es más ventajoso es en programación de sistemas embebidos. Los sistemas embebidos, como los microcontroladores utilizados en electrodomésticos, vehículos y dispositivos médicos, tienen recursos limitados en términos de memoria y potencia de procesamiento. El ensamblador permite optimizar el uso de estos recursos al escribir código más eficiente y específico para el hardware.

3. Explicación del código en ensamblador

- **Línea 1: MOV AX, 5**
  - Esta instrucción mueve el valor 5 al registro AX.
  - **Acción:** AX = 5
- **Línea 2: MOV BX, 10**
  - **Explicación:** Esta instrucción mueve el valor 10 al registro BX.
  - **Acción:** BX = 10
- **Línea 3: ADD AX, BX**

    - Esta instrucción suma el valor del registro BX al registro AX.
    - **Acción:** AX = AX + BX (AX = 5 + 10 = 15)
- **Línea 4: MOV CX, AX**
    - Esta instrucción mueve el valor del registro AX al registro CX.
    - **Acción:** CX = AX (CX = 15)

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

Funcionamiento de los Compiladores

Un **compilador** es un programa que traduce código escrito en un lenguaje de programación de alto nivel (como C++, Java o Python) a un lenguaje de máquina que pueda ser ejecutado por una computadora.

Análisis Léxico:

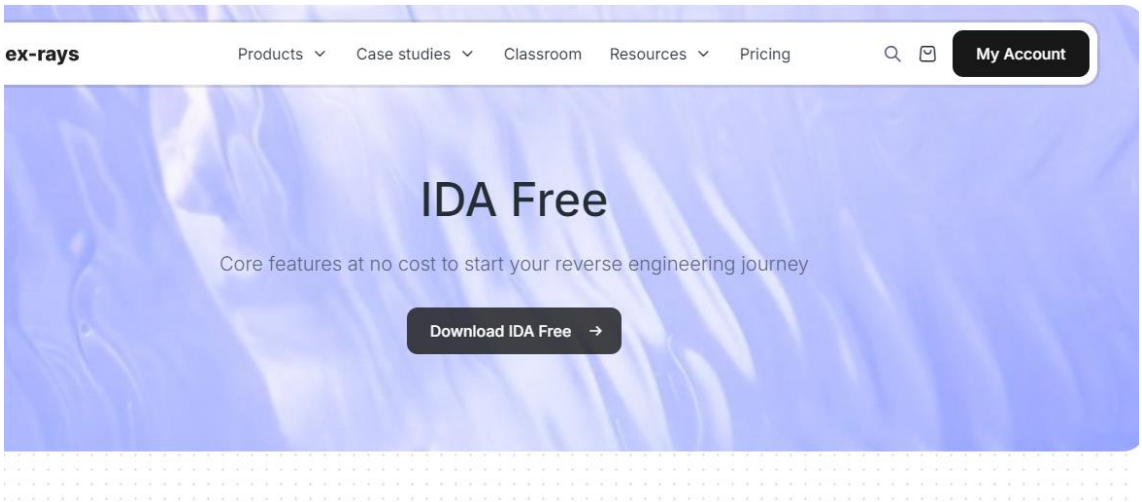
    - **Descripción:** Esta es la primera fase en la que el compilador toma el código fuente y lo divide en unidades básicas llamadas tokens. Los tokens son secuencias de caracteres que tienen un significado colectivo (como palabras clave, identificadores, operadores y delimitadores).
    - **Herramienta:** Analizador léxico (scanner).
2. **Análisis Sintáctico:**
    - **Descripción:** En esta fase, el compilador utiliza los tokens generados en el análisis léxico para construir una estructura denominada árbol sintáctico (o árbol de análisis). El árbol sintáctico muestra la estructura jerárquica del código fuente según las reglas gramaticales del lenguaje.
    - **Herramienta:** Analizador sintáctico (parser).
3. **Análisis Semántico:**
    - **Descripción:** Aquí, el compilador verifica si las construcciones del lenguaje en el árbol sintáctico tienen un significado lógico y válido. Por ejemplo, comprueba que las variables hayan sido declaradas antes de su uso y que los tipos de datos sean compatibles en las operaciones.
    - **Herramienta:** Analizador semántico.
4. **Optimización:**
    - **Descripción:** En esta fase, el compilador mejora el código intermedio para hacerlo más eficiente en términos de velocidad de ejecución y uso de recursos. Las optimizaciones pueden incluir la eliminación de código muerto, la simplificación de expresiones y la mejora de la asignación de registros.
    - **Herramienta:** Optimizador de código.
5. **Generación de Código Intermedio:**
    - **Descripción:** El compilador traduce el árbol sintáctico a un código intermedio que es más abstracto que el código máquina pero más específico que el código fuente. Este código intermedio facilita las optimizaciones y la generación final del código máquina.
    - **Herramienta:** Generador de código intermedio.
6. **Generación de Código Máquina:**

- **Descripción:** La fase final del compilador traduce el código intermedio optimizado en código máquina específico para la arquitectura del sistema en el que se ejecutará el programa. Este código puede ser directamente ejecutado por el procesador.
- **Herramienta:** Generador de código.

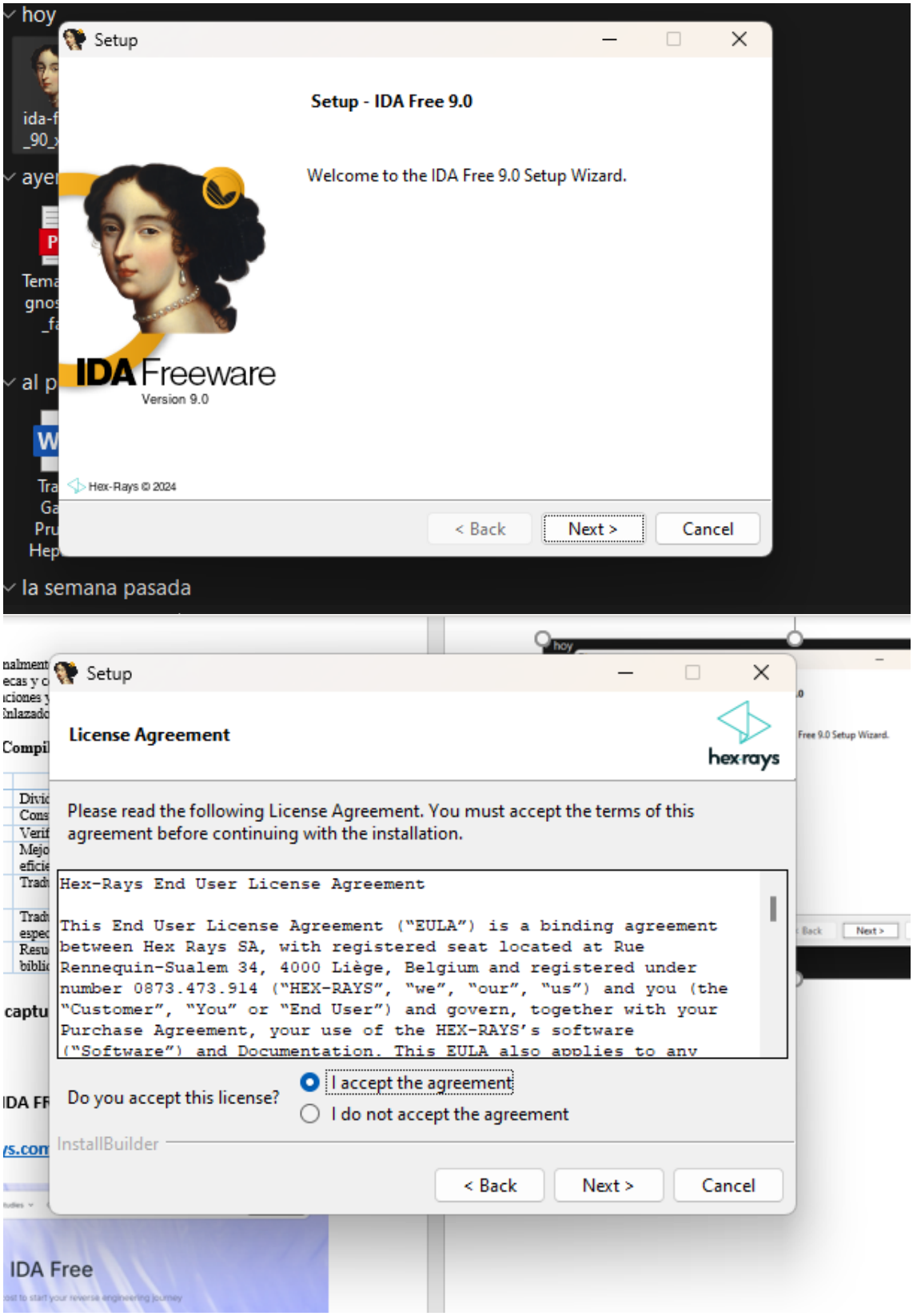
7. **Enlace y Ensamblaje:**
   - **Descripción:** Finalmente, el código máquina puede requerir ser enlazado con otras bibliotecas y componentes. Este proceso asegura que todas las referencias a funciones y variables sean resueltas correctamente.
   - **Herramienta:** Enlazador y ensamblador.

5) Realizar sus propias capturas de pantalla del siguiente procedimiento:

Paso 1:
Descargar el software IDA FREE el cual lo podrá a hacer del siguiente
enlace: https://hex-rays.com/ida-free/

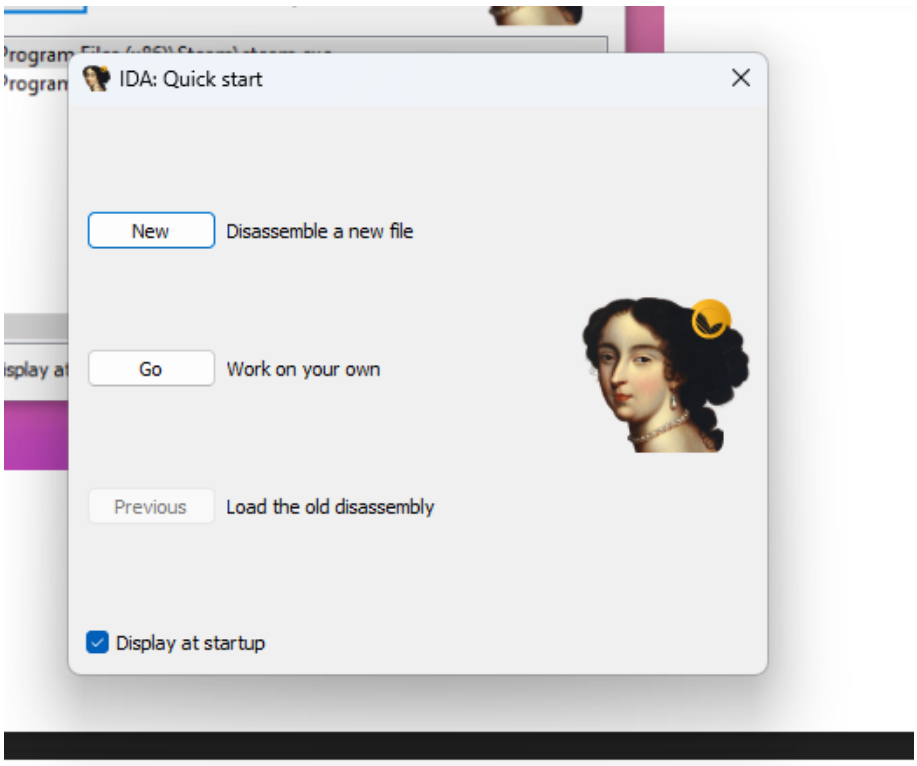


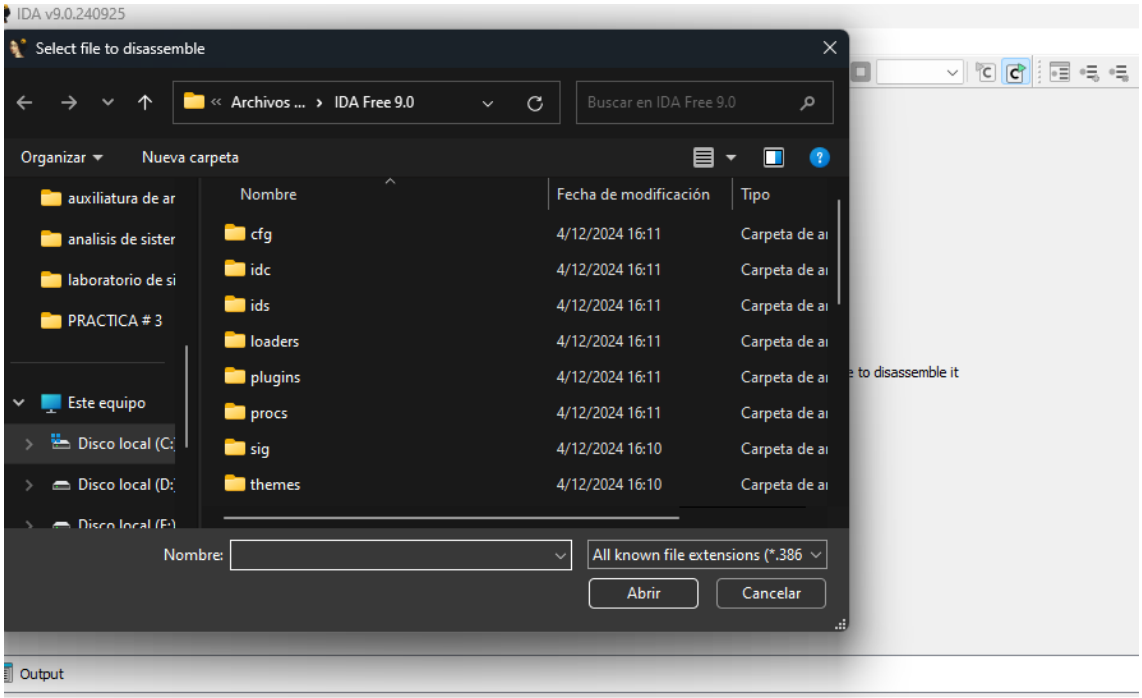
Paso 2: instalación



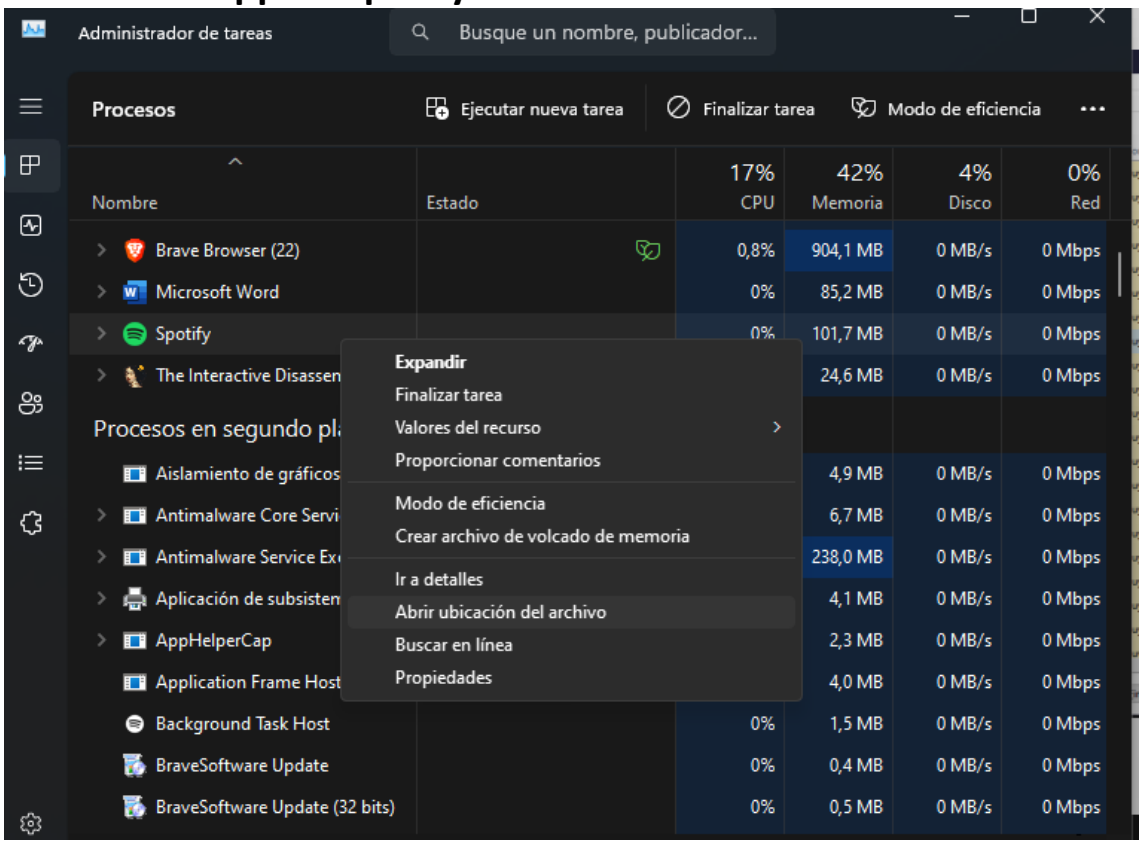
Paso 3:

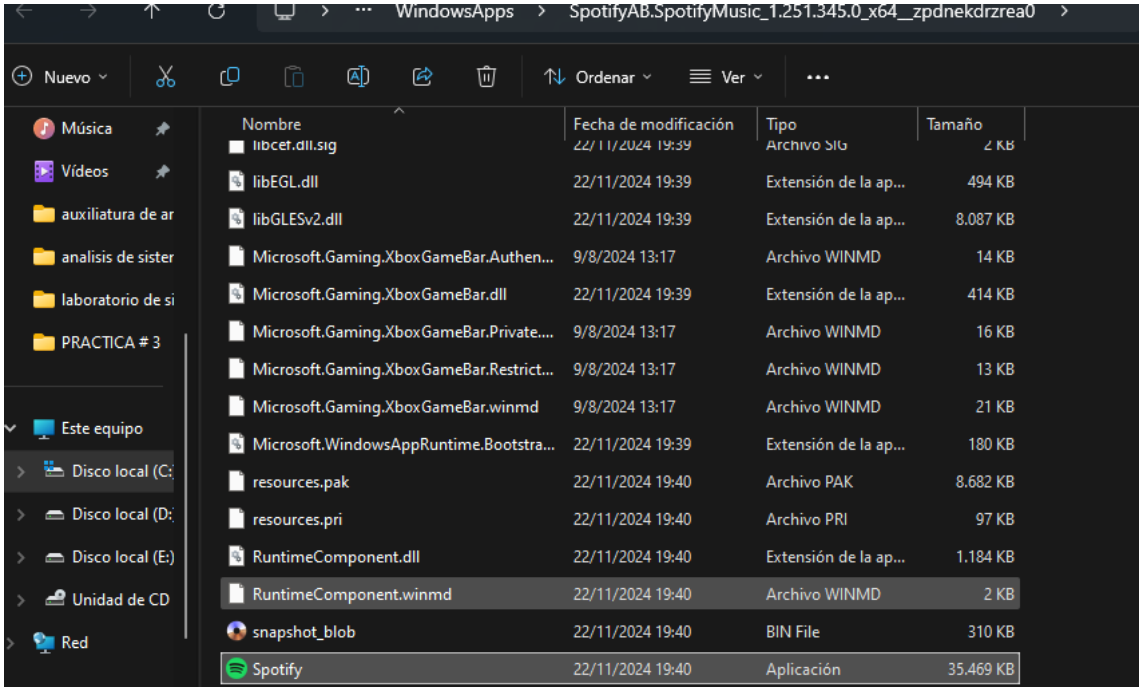
Procederemos a abrir un servicio en Windows



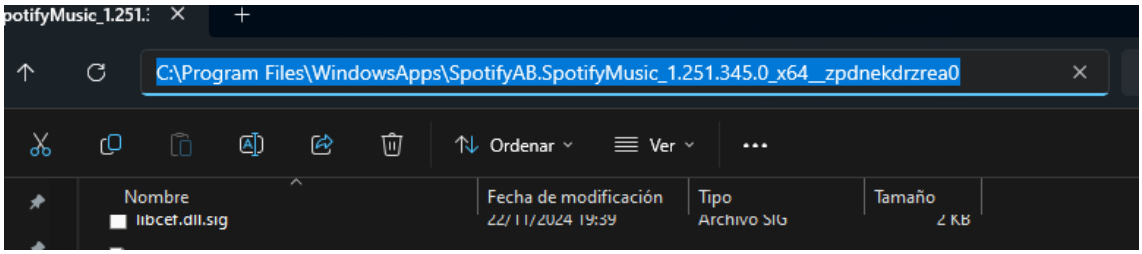


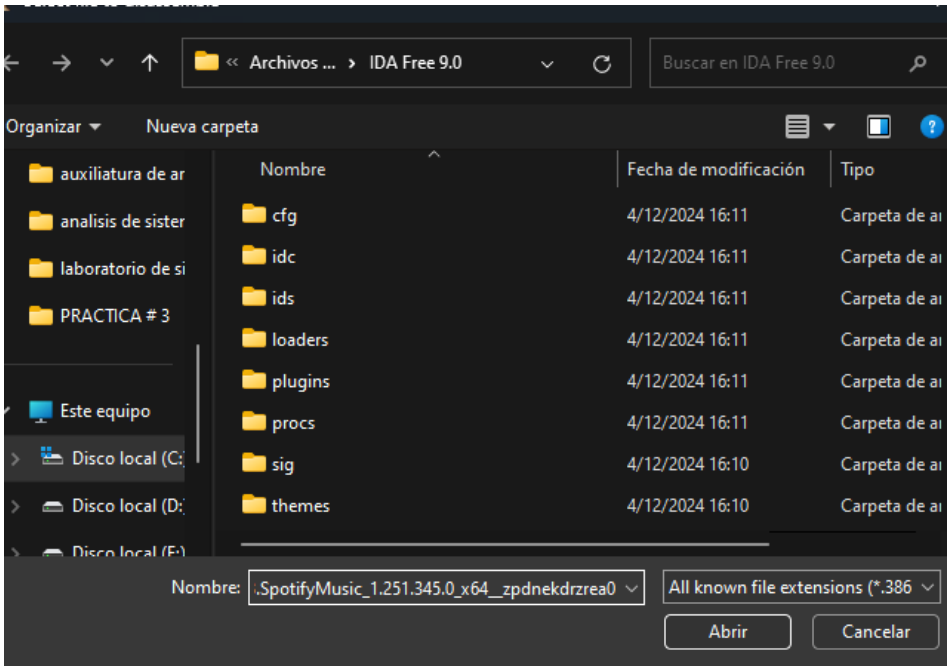
Se abrirá la app de Spotify

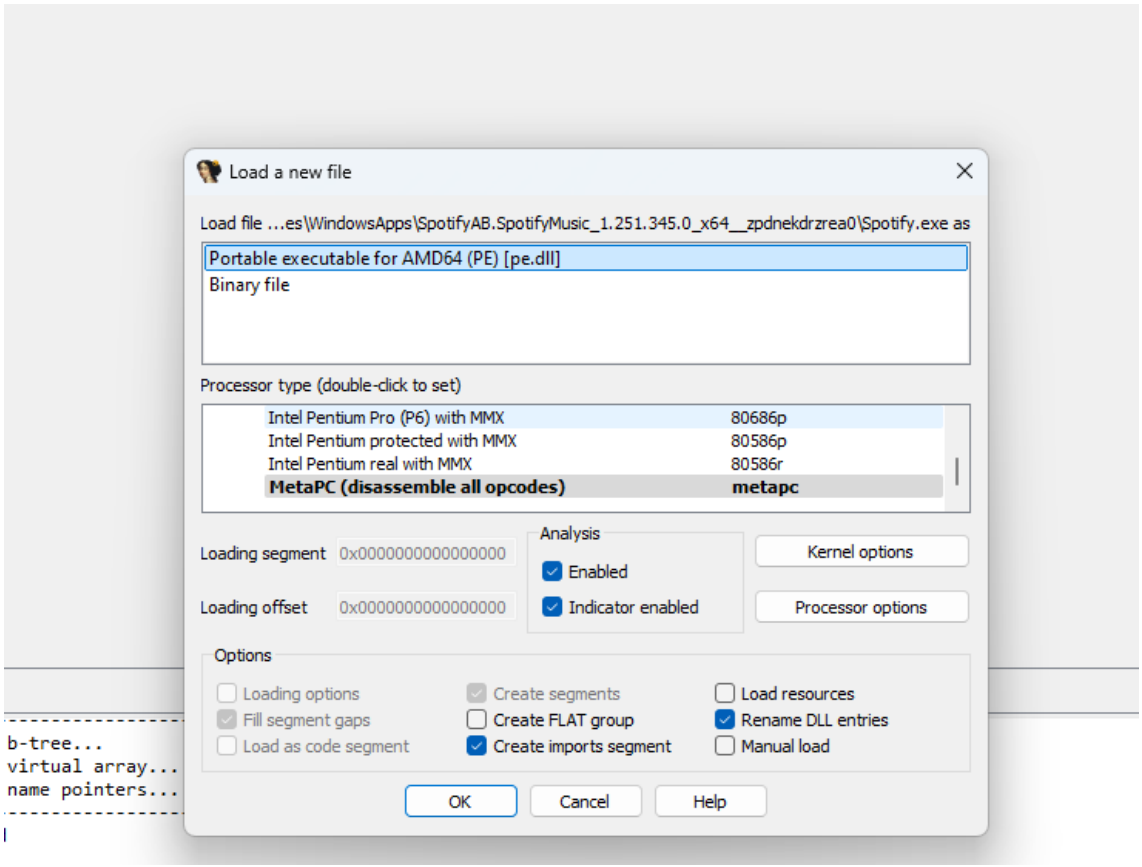


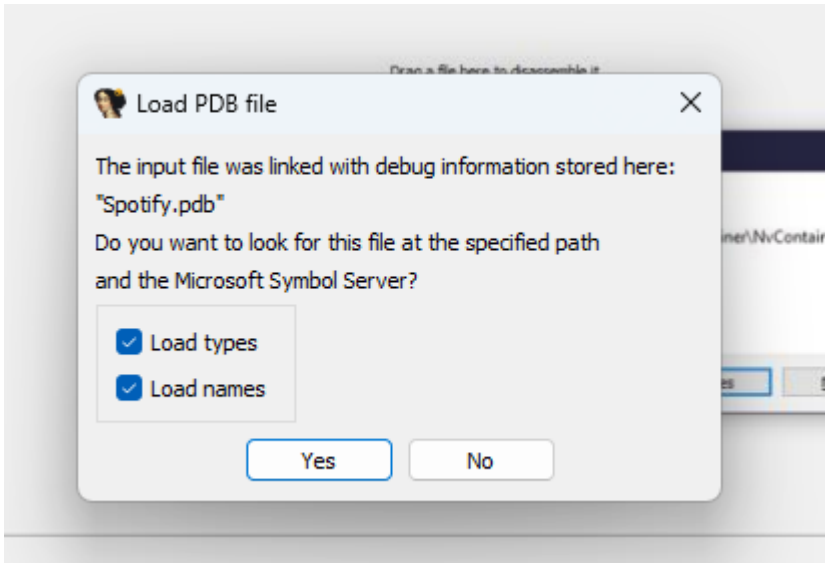


Ruta del archivo copiada






Load a new file
Load file ...es\WindowsApps\SpotifyAB.SpotifyMusic_1.251.345.0_x64__zpdnekdrzrea0\Spotify.exe as
Portable executable for AMD64 (PE) [pe.dll]
Binary file
Processor type (double-click to set)
Intel Pentium Pro (P6) with MMX 80686p
Intel Pentium protected with MMX 80586p
Intel Pentium real with MMX 80586r
MetaPC (disassemble all opcodes) metapc
Loading segment 0x0000000000000000
Loading offset 0x0000000000000000
Analysis
Enabled
Indicator enabled
Kernel options
Processor options
Options
Loading options
Fill segment gaps
Load as code segment
Create segments
Create FLAT group
Create imports segment
Load resources
Rename DLL entries
Manual load
OK
Cancel
Help
b-tree...
virtual array...
name pointers...


Load PDB file
The input file was linked with debug information stored here:
"Spotify.pdb"
Do you want to look for this file at the specified path
and the Microsoft Symbol Server?
Load types
Load names
Yes
No

Paso 4:

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