


<u>UNIVERSIDAD AUTÓNOMA “TOMAS FRÍAS”</u> <u>CARRERA DE INGENIERÍA DE SISTEMAS</u>				
Materia:	Arquitectura de computadoras (SIS-522)			
Docente:	Ing. Gustavo A. Puita Choque			N° Práctica
Auxiliar:	Univ. Aldrin Roger Perez Miranda			9
Estudiante	Univ. Dafne Rosario Tapia Parisaca			
20/11/2024	Fecha publicación			
06/12/2024	Fecha de entrega			
Grupo:	1	Sede	Potosí	

Responda las siguientes preguntas de **MANERA CONCISA**

LAS RESPUESTAS DE MANERA DIGITAL en formato .pdf

PRÁCTICA ANULADA
COPIA DE
JHON JAIRO GOMEZ CORDOVA
MARTA ISIDORA TACURI
Chicchi_Luis_Fernando

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

En ensamblador, el *stack* es una estructura de datos que sigue el principio de **LIFO** (Last In, First Out). Se usa para almacenar temporalmente información como direcciones de retorno de funciones, variables locales y registros.

Como Se Usa:

1. **Llamadas a funciones:** Cuando se llama a una función, la dirección de retorno se guarda en el stack. Cuando la función termina, se recupera esta dirección para continuar la ejecución.
2. **Almacenamiento de registros:** Se pueden guardar valores de registros en el stack para preservarlos durante las operaciones.
3. **Instrucciones comunes:**
 - ✓ **push:** Añade un valor al stack.
 - ✓ **pop:** Extrae un valor del stack.

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

El ensamblador es más ventajoso que los lenguajes de alto nivel en escenarios de **sistemas embebidos** o **tiempo real** con recursos limitados. En estos casos, permite:

1. **Control preciso del hardware:** Acceso directo a registros y memoria.
2. **Máximo rendimiento:** Elimina sobrecarga, optimizando la ejecución.
3. **Eficiencia de recursos:** Uso mínimo de memoria y almacenamiento.
4. **Requisitos de tiempo real:** Permite ejecutar tareas con alta precisión temporal.

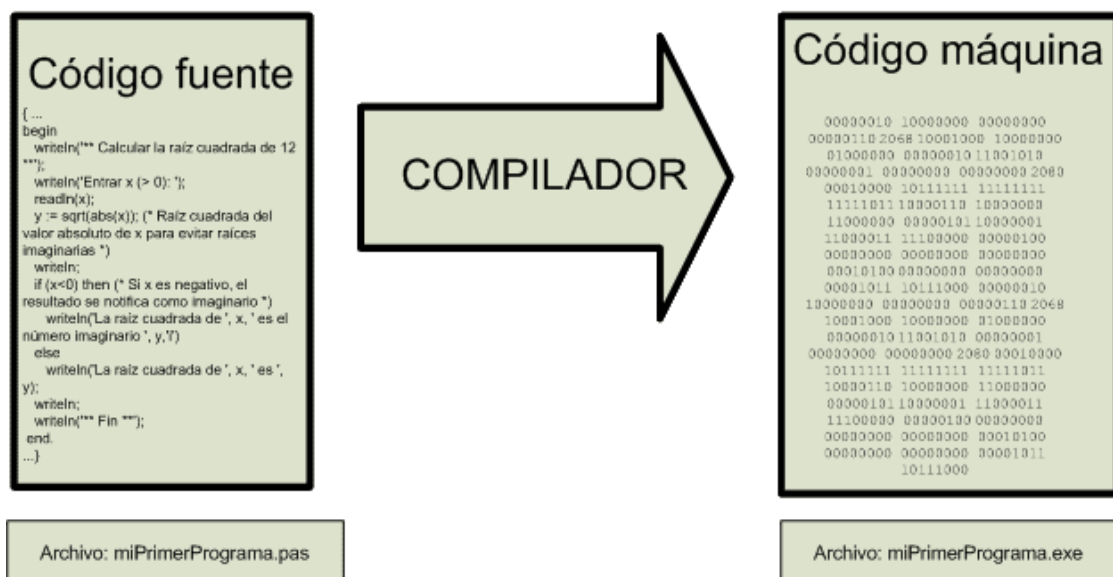
3) Explique cada línea del siguiente código del lenguaje ensamblador y diga que es lo que se está haciendo

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

Este código en lenguaje ensamblador realiza operaciones básicas de asignación y aritmética entre registros. A continuación, se explica cada línea:

1. **MOV AX, 5:**
 - Esta línea mueve el valor **5** al registro **AX**. El comando **MOV** se usa para transferir datos de un operando a otro. En este caso, **5** es un valor inmediato que se asigna al registro **AX**.
2. **MOV BX, 10:**
 - Similar a la línea anterior, esta instrucción mueve el valor **10** al registro **BX**. Ahora, **BX** contiene el valor 10.
3. **ADD AX, BX:**
 - Aquí, se realiza una operación de **suma** entre los valores contenidos en los registros **AX** y **BX**. El valor de **BX** (10) se suma al valor en **AX** (5). El resultado (15) se guarda en **AX**.
4. **MOV CX, AX:**
 - Esta línea mueve el valor de **AX** (que ahora es 15) al registro **CX**. Después de esta instrucción, **CX** tendrá el valor **15**.

4) Explique detalladamente cómo funcionan los compiladores



Un compilador convierte el código fuente de un lenguaje de alto nivel a un lenguaje de bajo nivel (como código máquina). Las etapas principales son:

1. **Análisis Léxico:** Descompone el código en tokens (palabras clave, operadores, etc.).
2. **Análisis Sintáctico:** Verifica que la estructura del código siga las reglas del lenguaje.
3. **Análisis Semántico:** Comprueba que el código sea lógicamente correcto (por ejemplo, tipos de datos adecuados).
4. **Generación de Código Intermedio:** Crea una representación más cercana al código máquina, pero independiente de la plataforma.
5. **Optimización del Código:** Mejora el código para hacerlo más eficiente.
6. **Generación de Código de Máquina:** Convierte el código intermedio en un código específico para la arquitectura del procesador.
7. **Ensamblado y Enlace:** Junte módulos y bibliotecas para crear un archivo ejecutable.

Cada fase es crucial para garantizar que el programa funcione correctamente en el sistema destino.

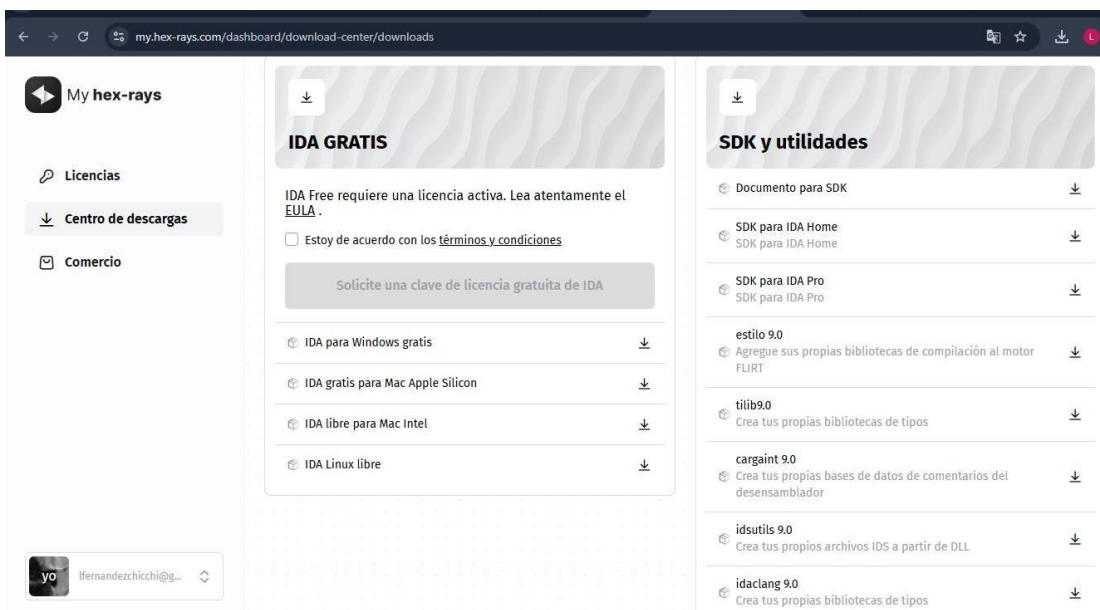
5) Realizar sus propias capturas de pantalla del siguiente procedimiento:

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

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:

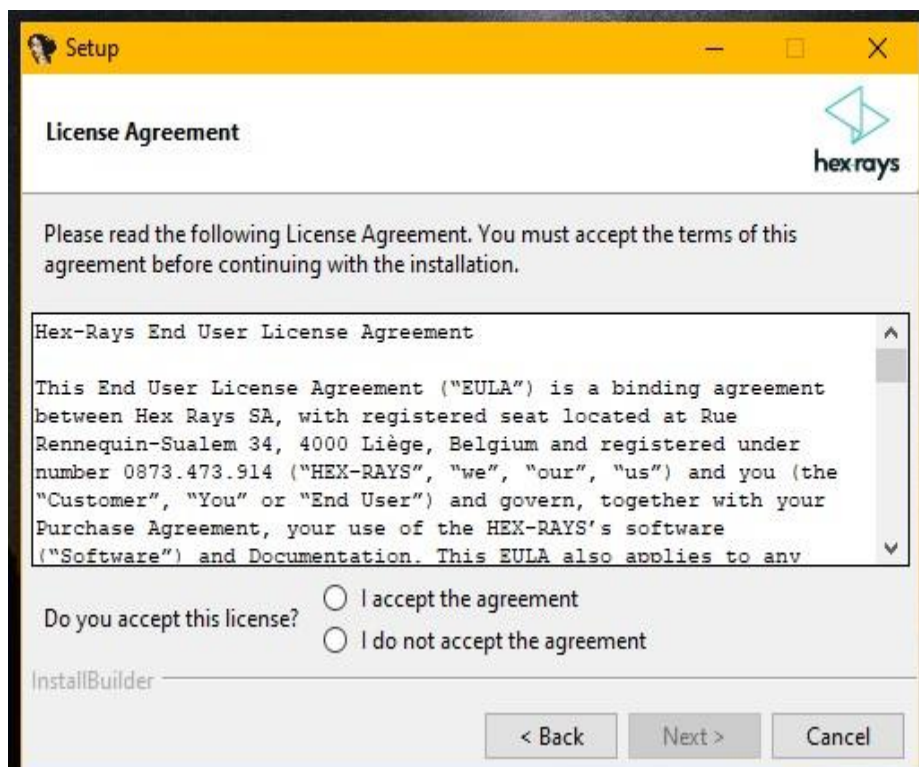
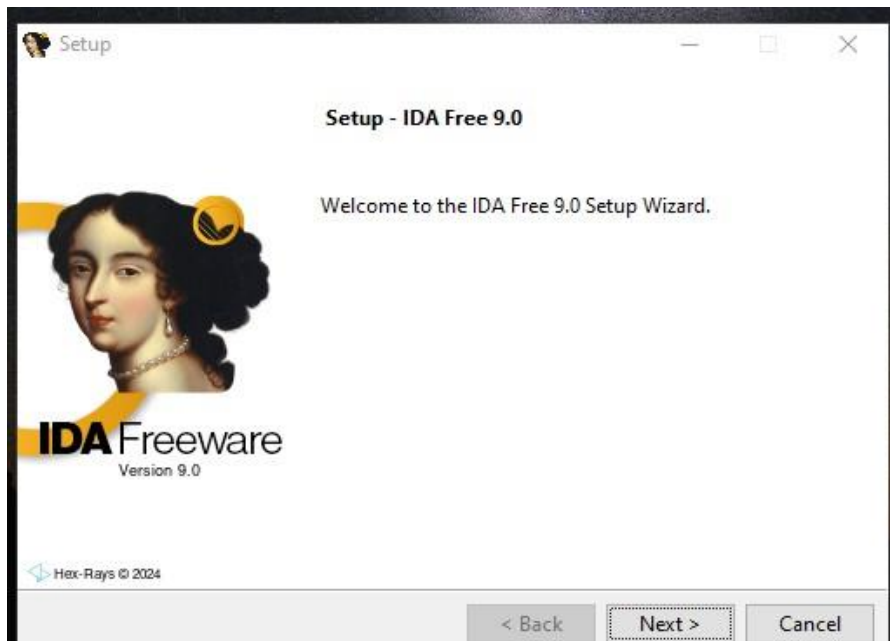
Descargar el software IDA FREE con todas las licencias disponibles.

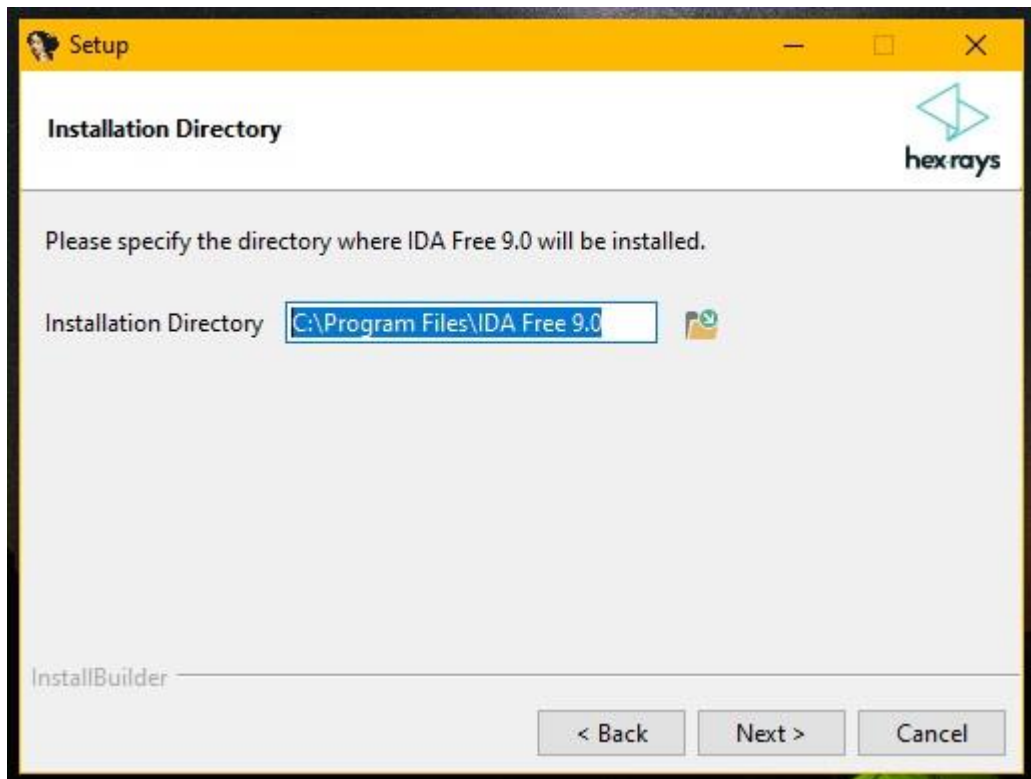


Paso 2:

Instalación

Seleccionar Next





Una vez descargado e instalado deberán abrir el ejecutable .exe

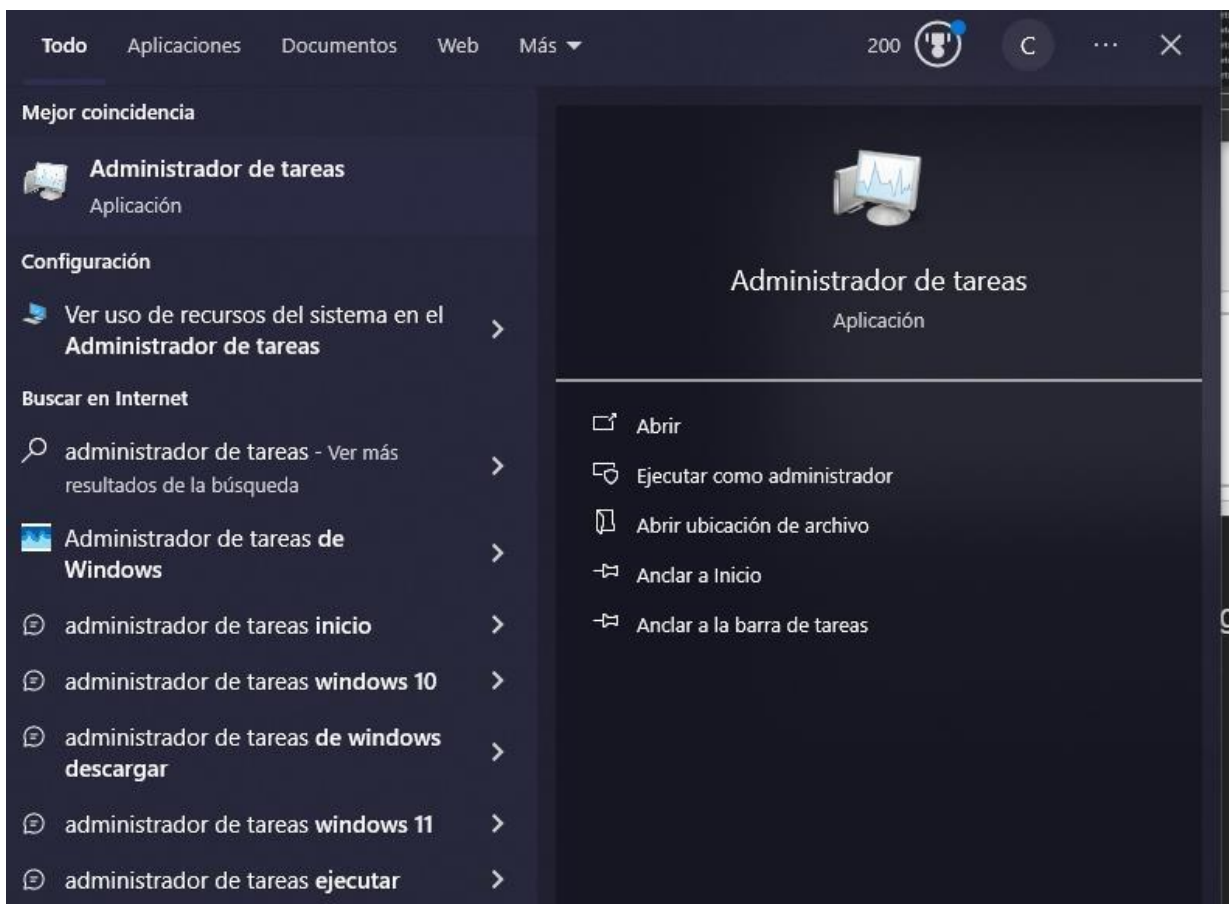


Paso 3:

Procederemos a abrir un servicio en Windows



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”

Administrador de tareas

Archivo Opciones Vista

Procesos Rendimiento Historial de aplicaciones Inicio Usuarios Detalles Servicios

Nombre	Estado	18% CPU	54% Memoria	12% Disco	0% Red
Aplicaciones (7)					
> Administrador de tareas		4,4%	19,9 MB	0 MB/s	0 Mbps
> Explorador de Windows (5)		0%	45,4 MB	0 MB/s	0 Mbps
> Google Chrome (18)		0%	1.096,6 MB	0 MB/s	0 Mbps
> Microsoft Word (2)		0,9%	76,9 MB	0 MB/s	0 Mbps
> Reproductor multimedia (2)		0,6%	108,2 MB	0 MB/s	0 Mbps
> The Interactive Disassembler		0%	35,8 MB	0 MB/s	0 Mbps
> Warcraft III (32 bits)		0,9%	3,8 MB	0 MB/s	0 Mbps
Procesos en segundo plano (76)					
> 64-bit Synaptics Pointing Enhanc...		0%	0,1 MB	0 MB/s	0 Mbps
AggregatorHost.exe		0%	0,5 MB	0 MB/s	0 Mbps
Aislamiento de gráficos de disp...		4,3%	8,3 MB	0 MB/s	0 Mbps
> AMD External Events Service M...		0%	0 MB	0 MB/s	0 Mbps
> Antimalware Core Service		0%	0 MB	0 MB/s	0 Mbps

Menos detalles Finalizar tarea

Administrador de tareas

Archivo Opciones Vista

Procesos Rendimiento Historial de aplicaciones Inicio Usuarios Detalles Servicios

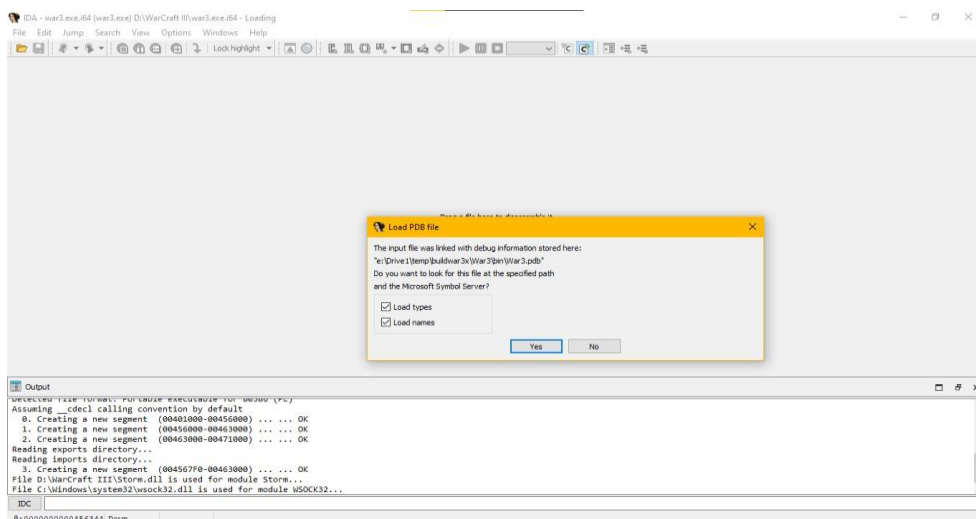
Nombre	Estado	33% CPU	54% Memoria	9% Disco	0% Red
Aplicaciones (7)					
> Administrador de tareas		1,8%	20,1 MB	0 MB/s	0 Mbps
> Explorador de Windows (5)		1,9%	45,6 MB	0 MB/s	0 Mbps
> Google Chrome (18)		0,6%	1.104,1 MB	0,1 MB/s	0 Mbps
> Microsoft Word (2)		0%	61,6 MB	0 MB/s	0 Mbps
> Reproductor multimedia (2)		0,4%	107,6 MB	0 MB/s	0 Mbps
> The Interactive Disassembler		0%	35,8 MB	0 MB/s	0 Mbps
> Warcraft III (32 bits)		0%	3,8 MB	0 MB/s	0 Mbps
Procesos en segundo plano (76)					
> 64-bit Synaptics Pointing Enhanc...		0%	0,1 MB	0 MB/s	0 Mbps
AggregatorHost.exe		0%	0,5 MB	0 MB/s	0 Mbps
Aislamiento de gráficos de disp...		6,5%	8,3 MB	0 MB/s	0 Mbps
> AMD External Events Service M...		0%	0 MB	0 MB/s	0 Mbps
> Antimalware Core Service		0%	0 MB	0 MB/s	0 Mbps

Menos detalles Finalizar tarea

Expandir
Finalizar tarea
Valores del recurso
Enviar comentarios
Crear archivo de volcado
Ir a detalles
Abrir ubicación del archivo
Buscar en línea
Propiedades

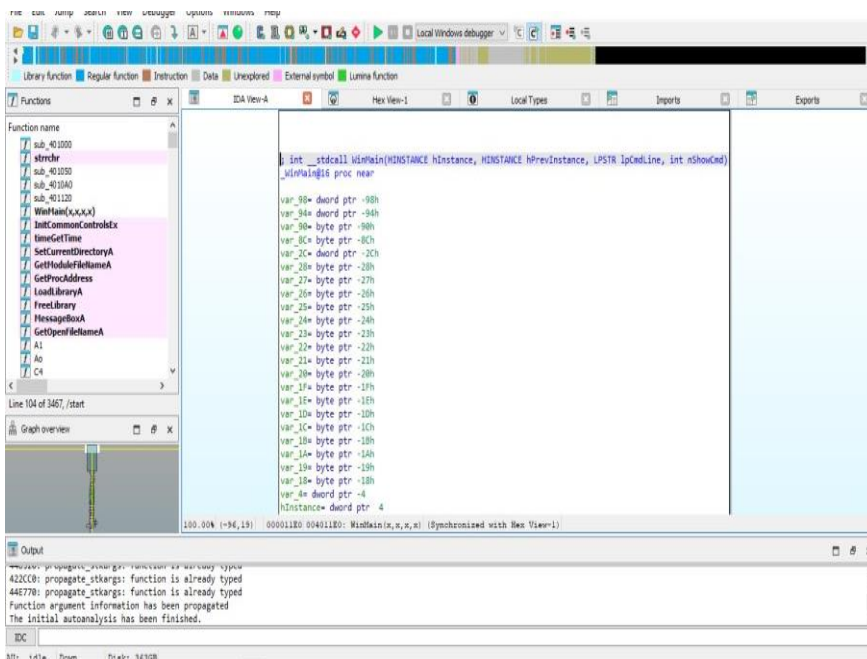
Una vez hecho esto se abrirá la ubicación del servicio
 Ahora se deberá copiar la ruta en donde esta este servicio el cual es en este caso

servicio a analizar. Dejaremos todo por definido y colocamos “ok”
Colocaremos “no”

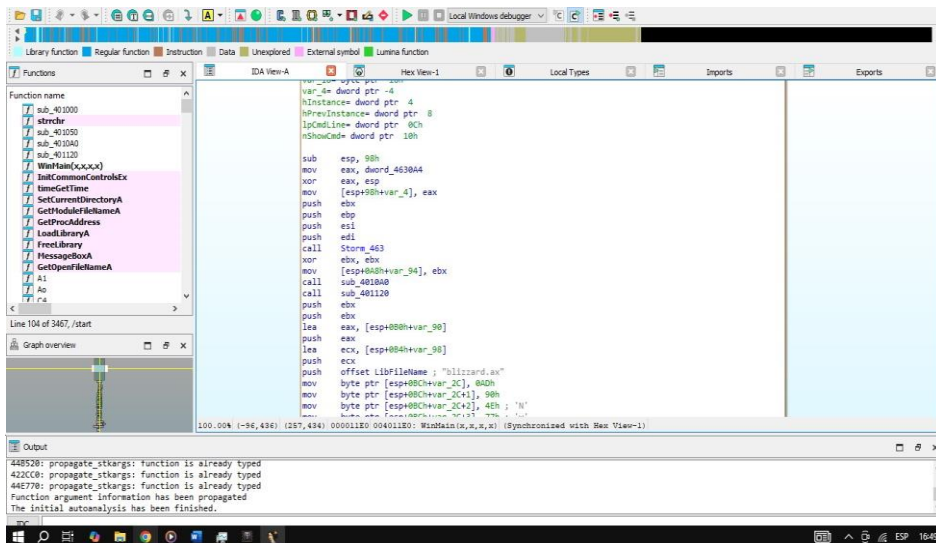
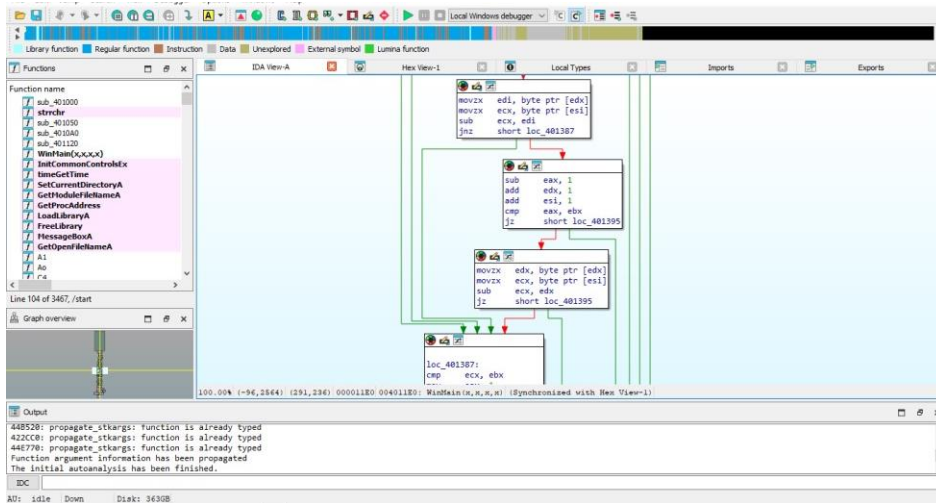


Paso 4:

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



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



Hasta este apartado debe generar sus propias capturas utilizando cualquier aplicación de su preferencia para observar el código desensamblado, pero *ya no debe ser Steam como en el ejemplo*