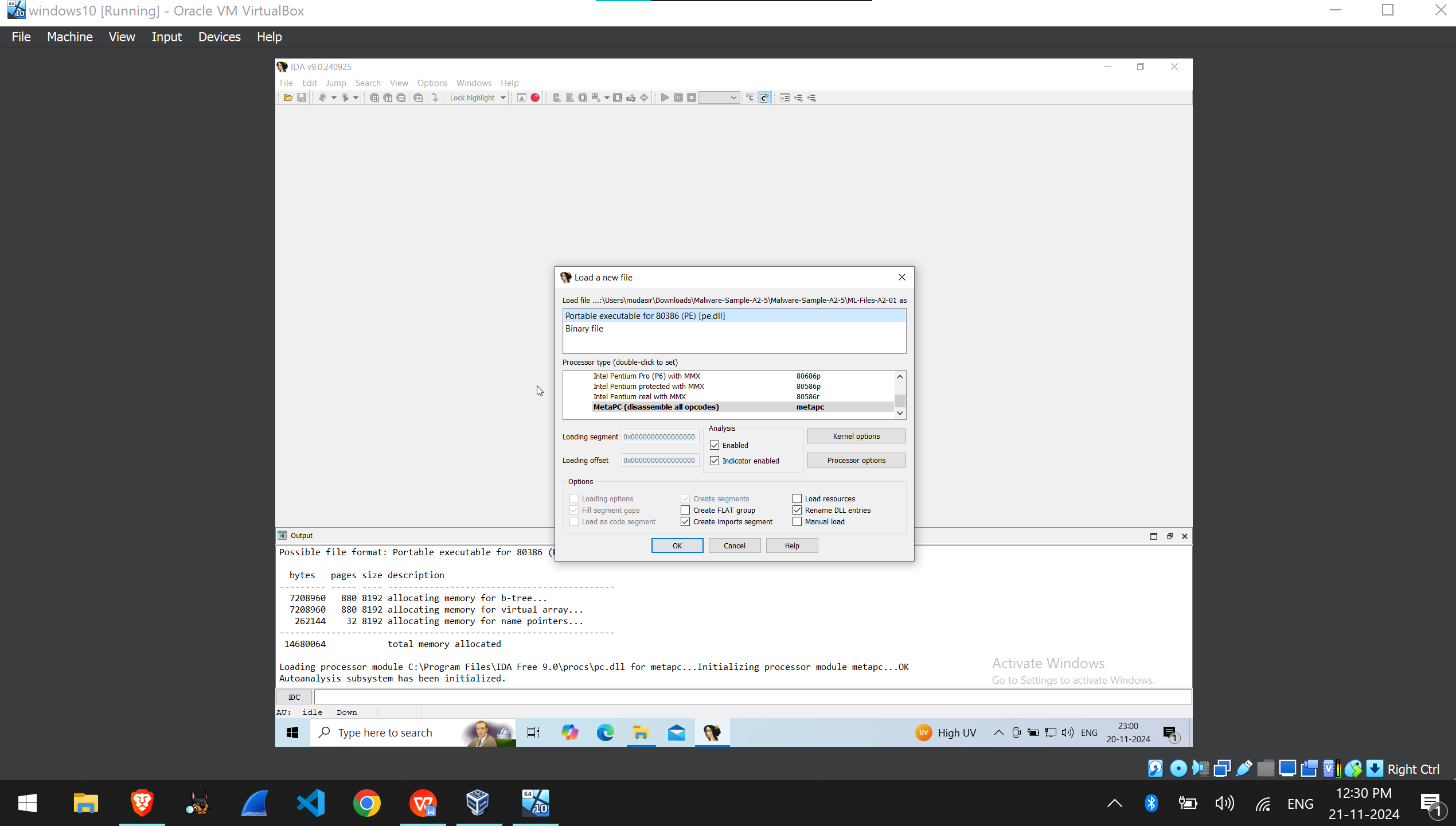
A.

#### a) File Format and Architecture

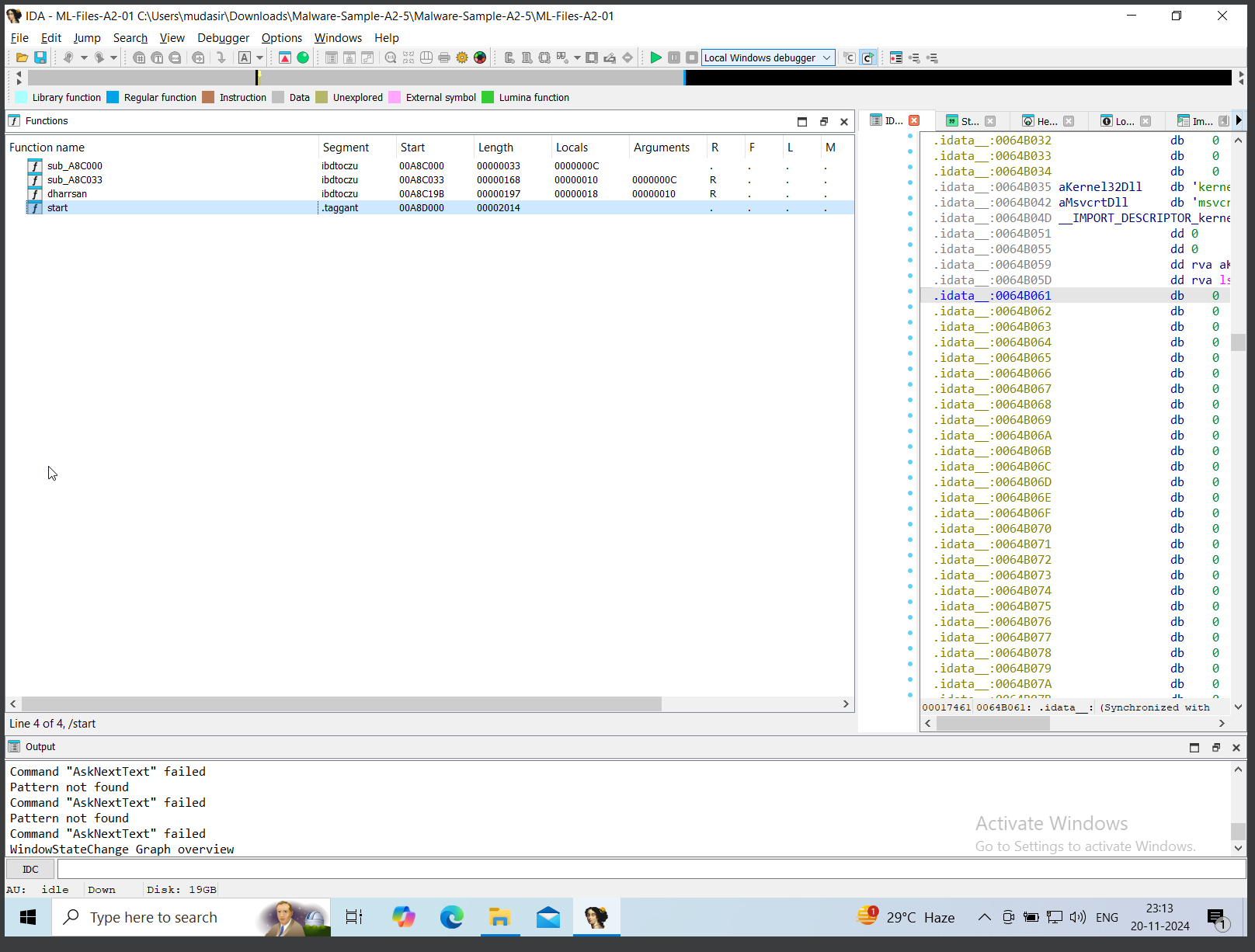
1. Open the binary file in IDA Pro.
2. IDA automatically detects the **file format** (e.g., PE for Windows, ELF for Linux) and **architecture** (e.g., x86, x64, ARM). This information is displayed in the **"Load a New File"** dialog.
3. To confirm:
   1. Go to **File > Load File > Parse C Header File**.
   2. Check **File > View > File Info** for architecture and format.

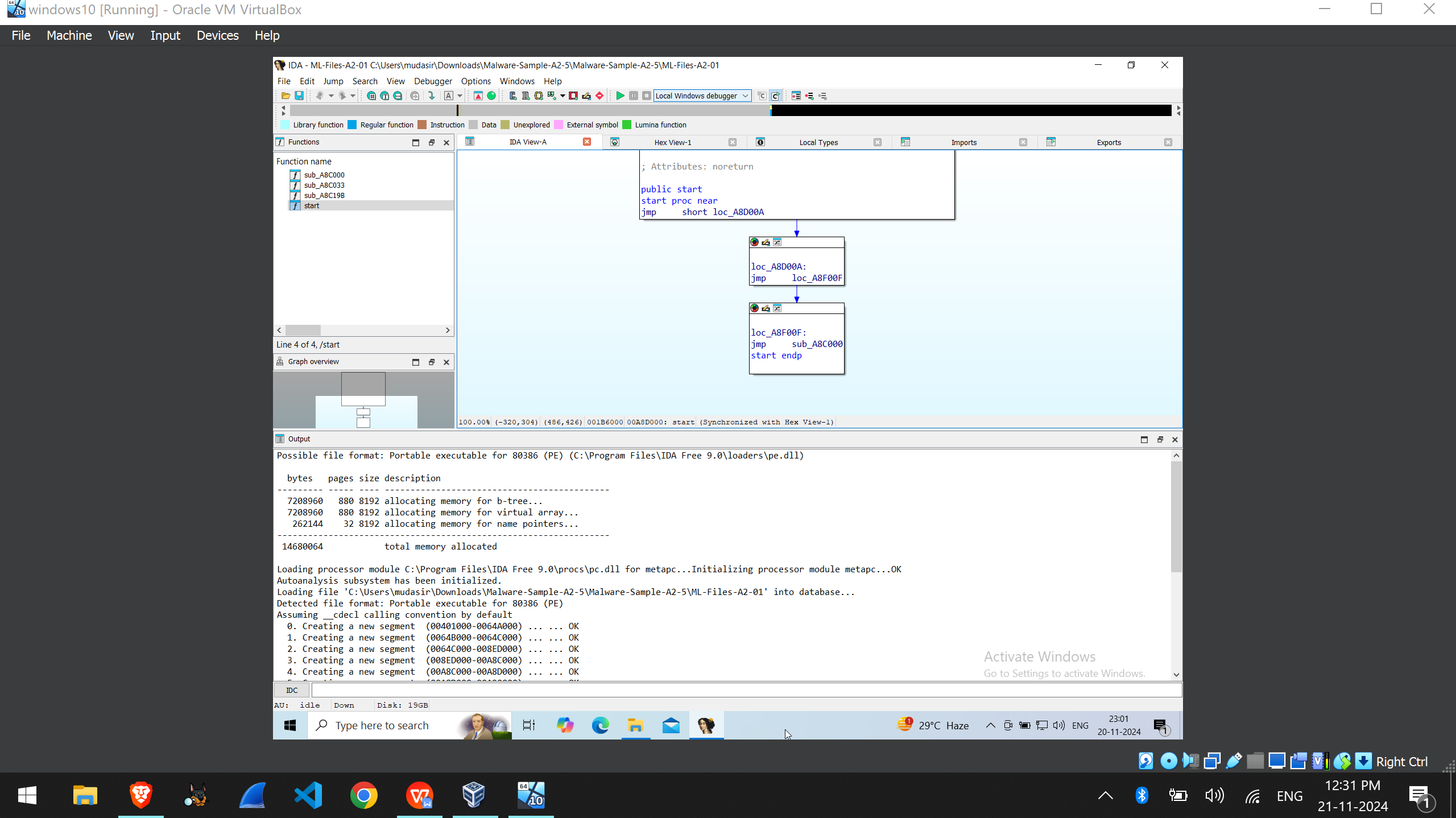


B.

#### b) Entry Point Address

1. The entry point is shown in the **"Segments" window** or the **Output Window** after loading.
   1. Alternatively, check the **"IDA View-A"** tab, where the main or \_start symbol may be near the entry point.

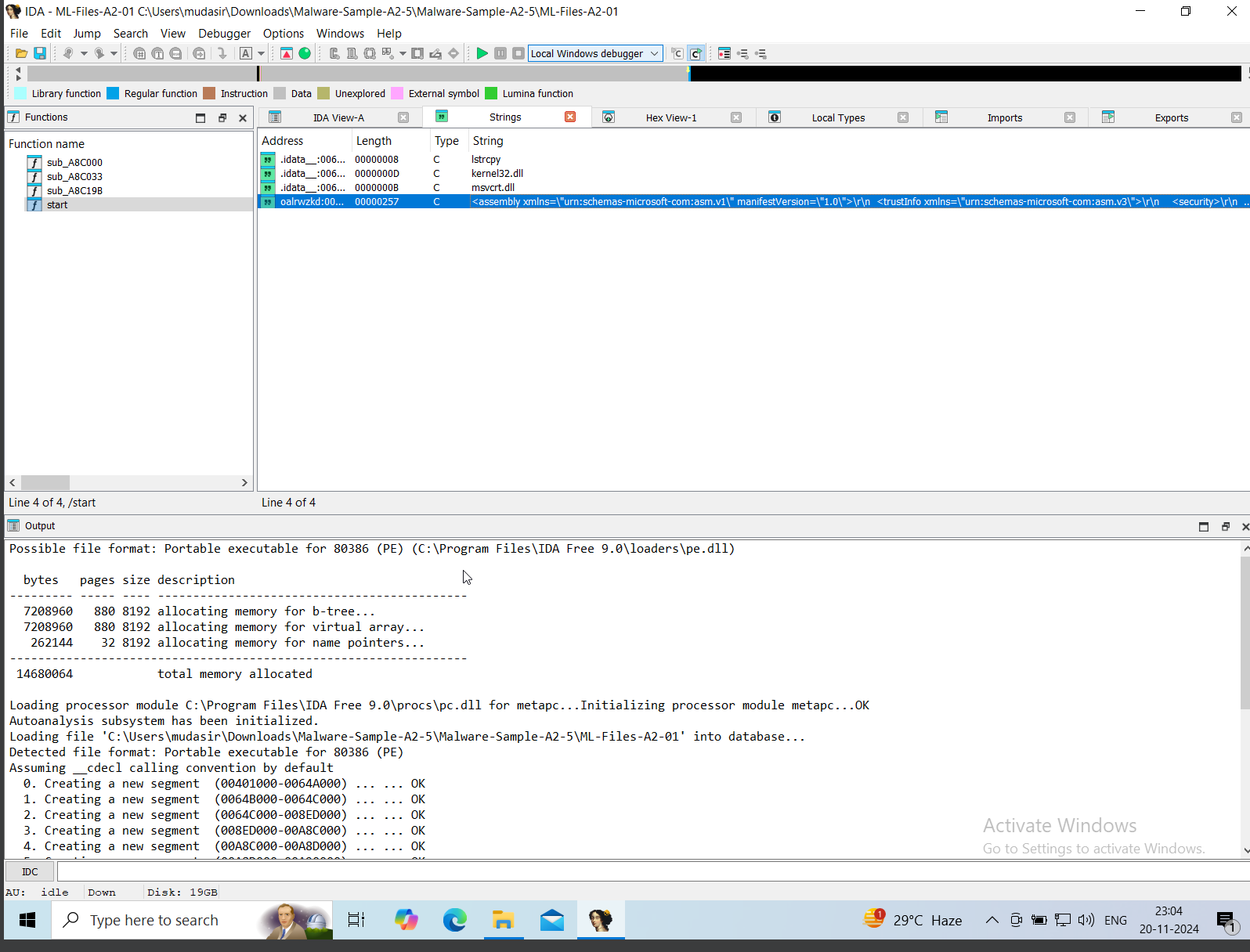


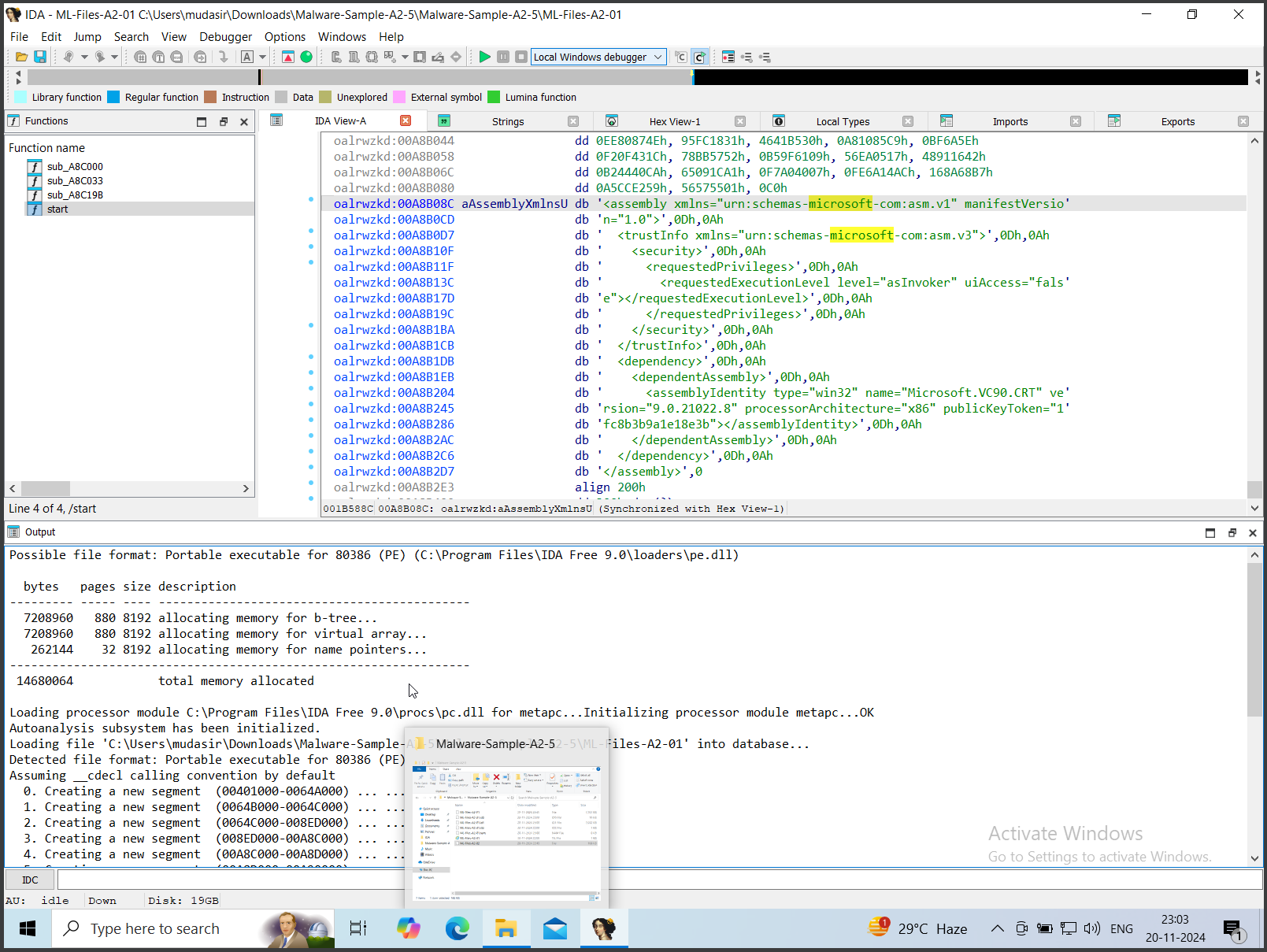


C.

#### c) Compiler Used

1. Look for the compiler metadata:
   1. Check the **Strings Window** (Shift + F12) for references to compilers (e.g., MSVC, GCC).
   2. Examine function prologues (specific patterns reveal compilers).

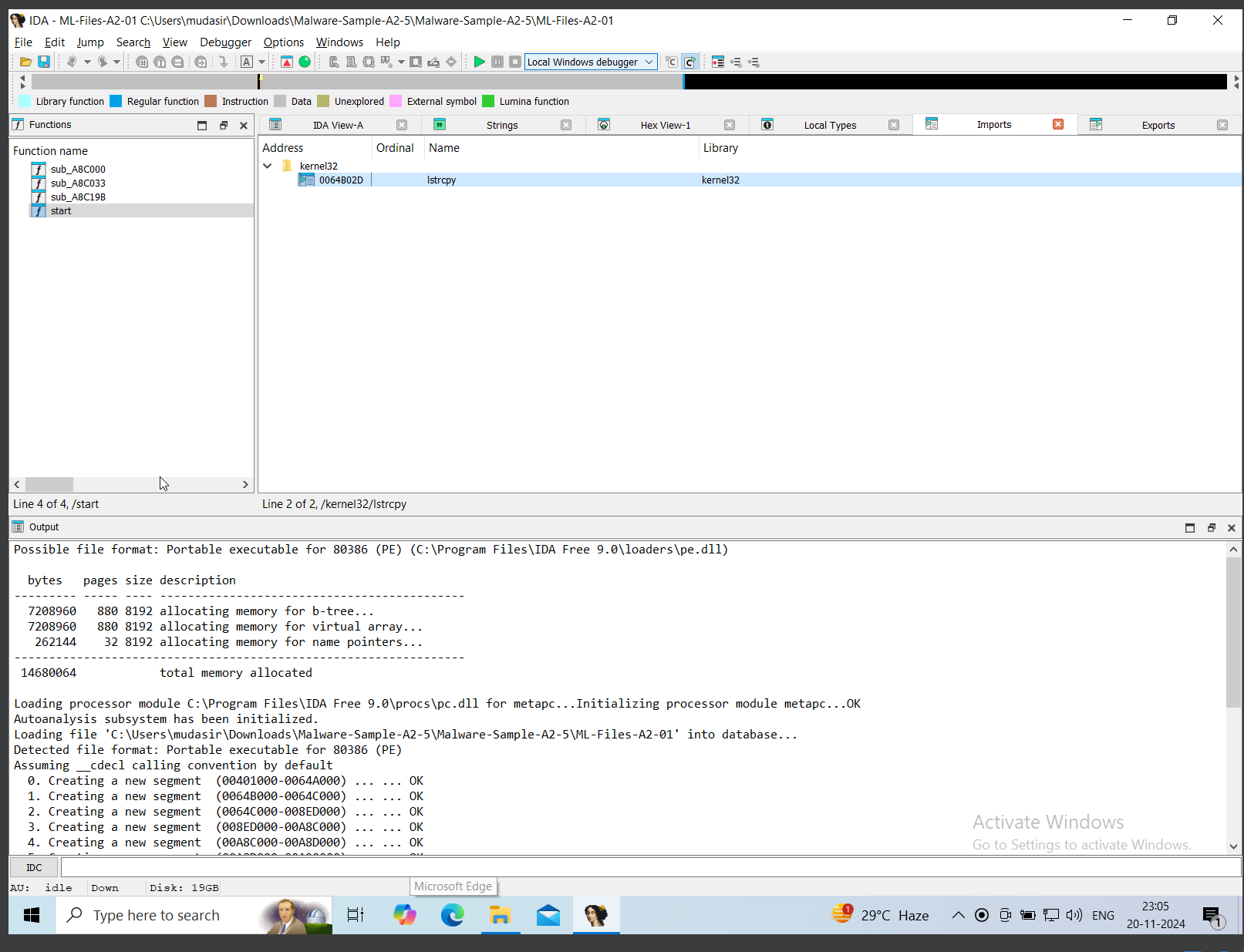




D.

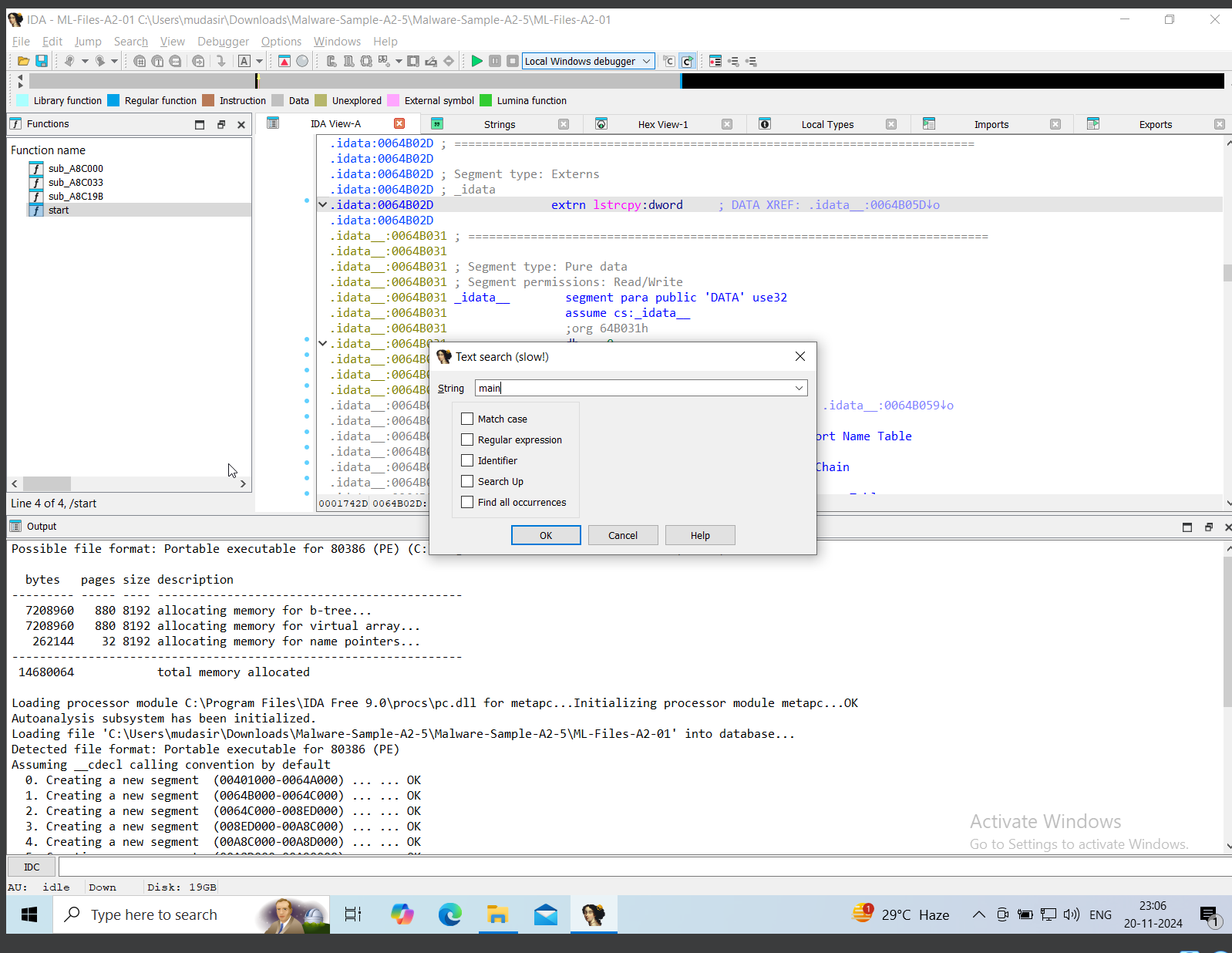
#### d) Import Table Contents

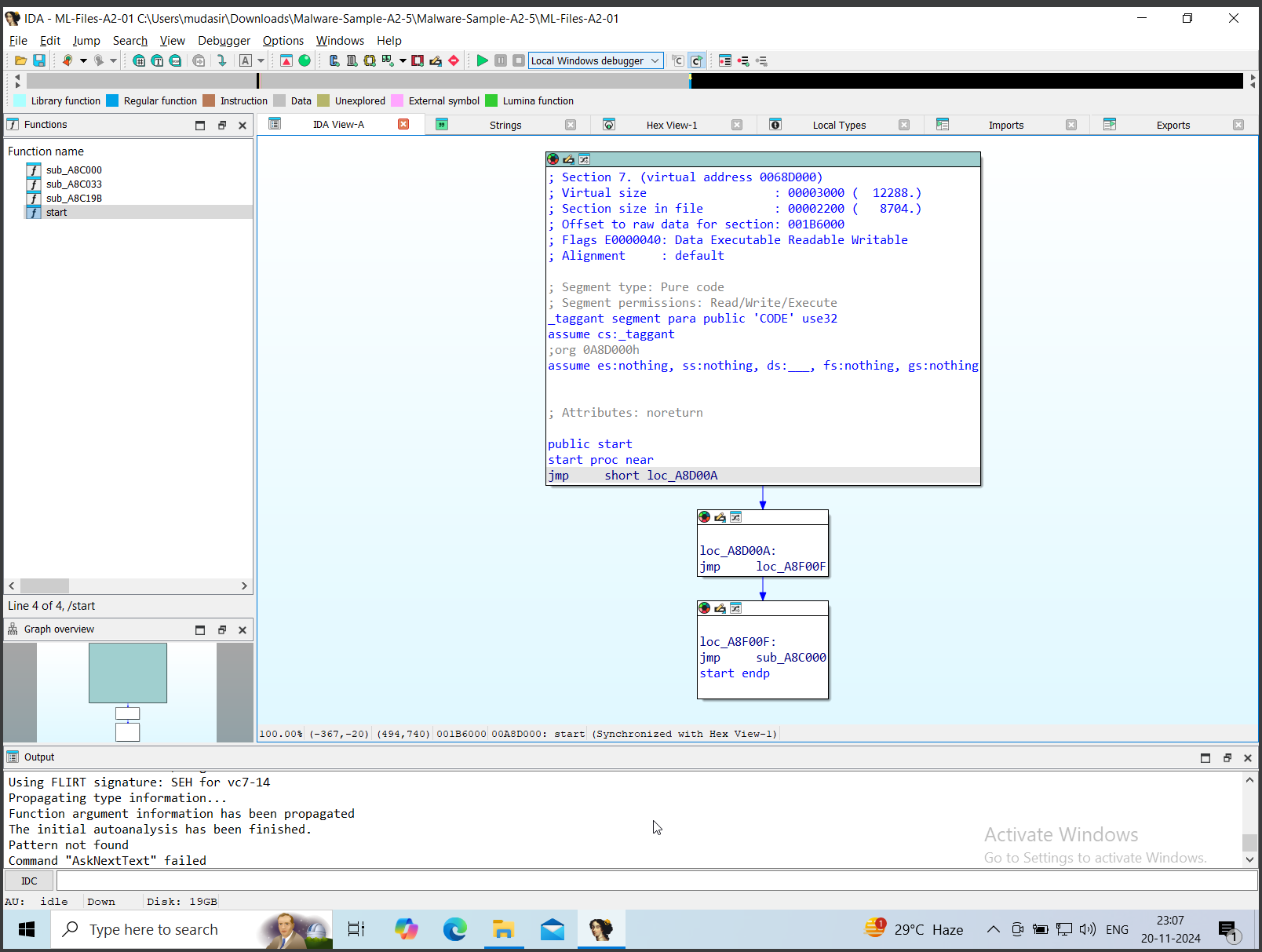
1. Open the **Imports window** by navigating to View > Open Subviews > Imports.
2. This displays libraries and their functions (e.g., kernel32.dll with functions like ReadFile).



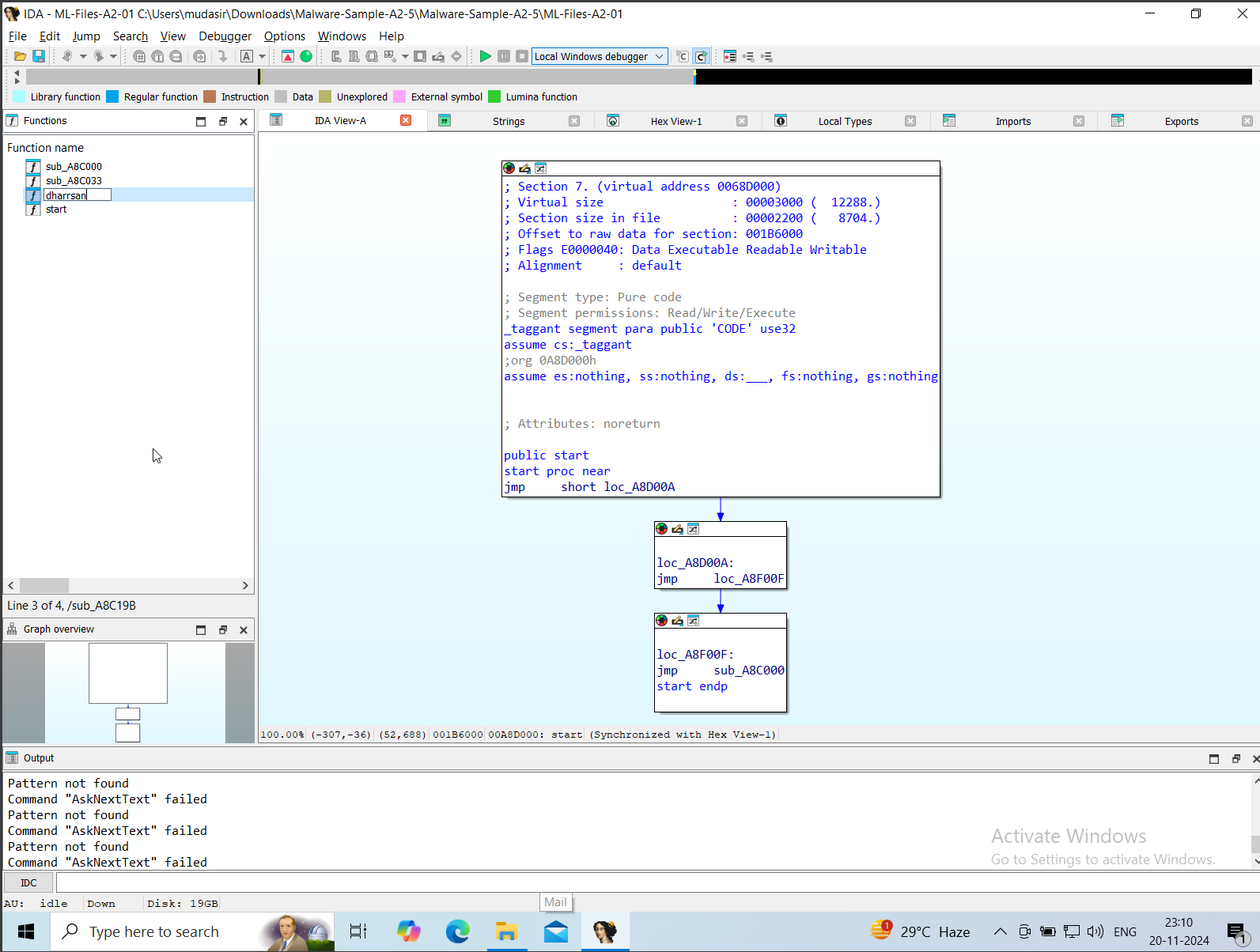
1. Main Function

A.

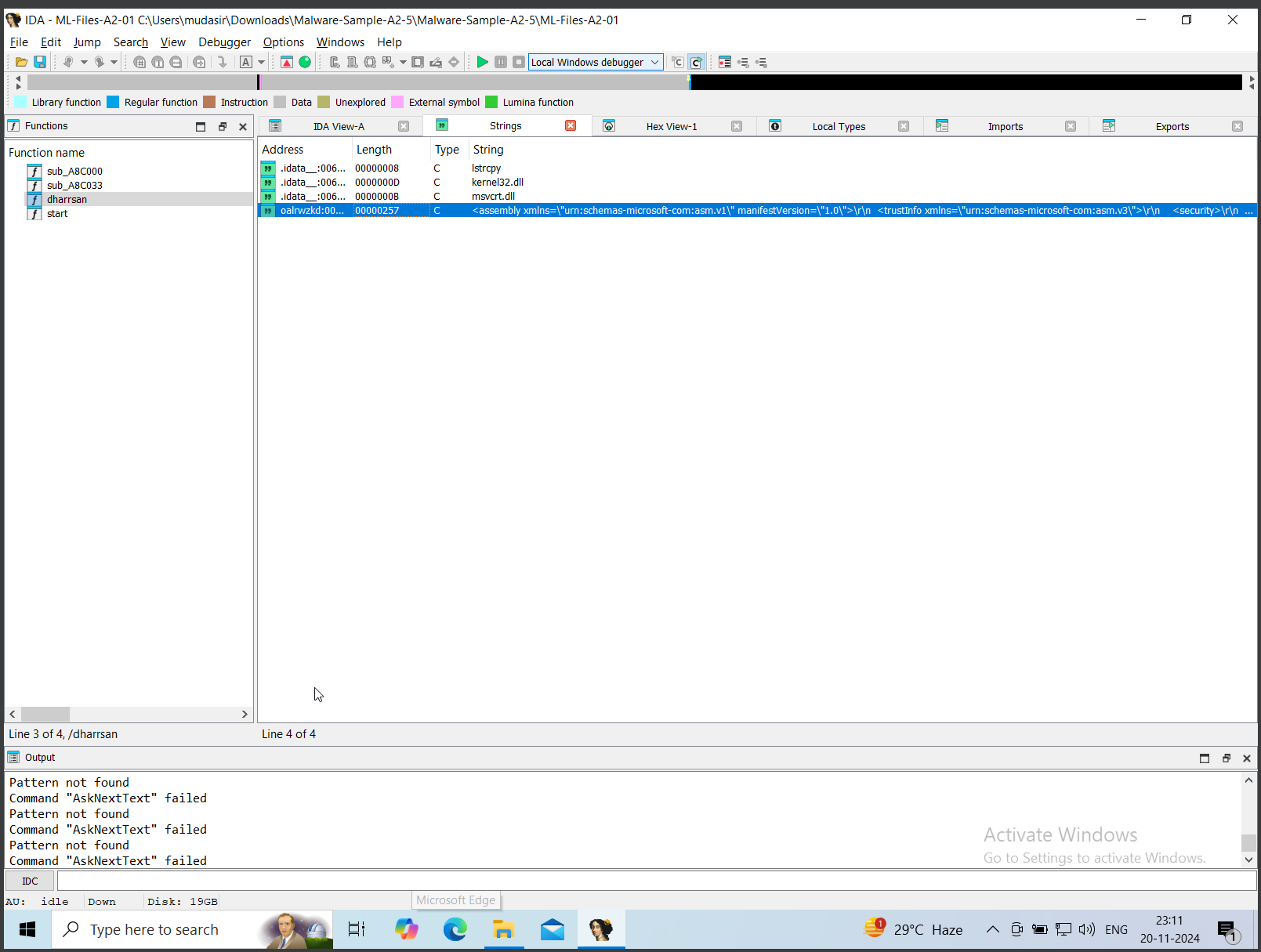


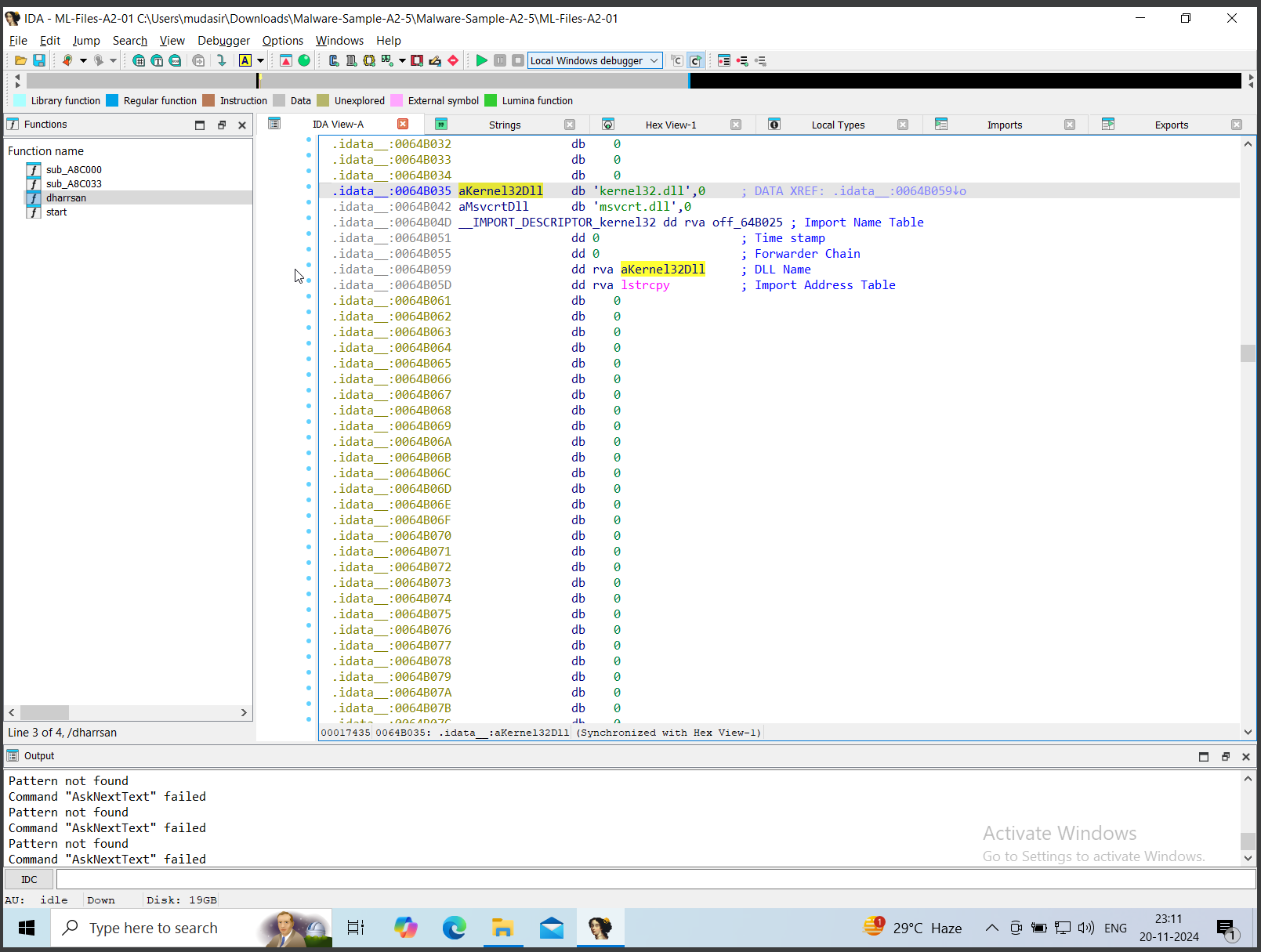


B.



C.

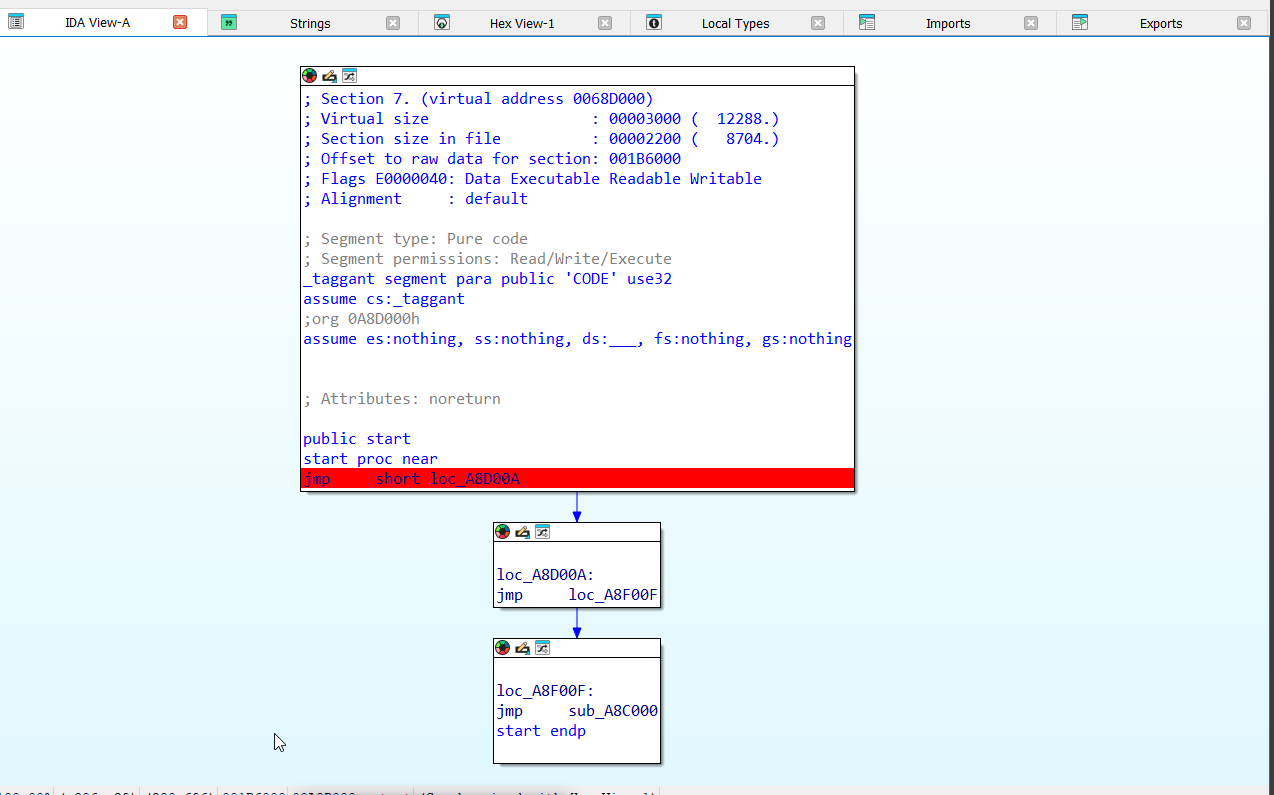




### ****3) Analyze the Program’s Control Flow****

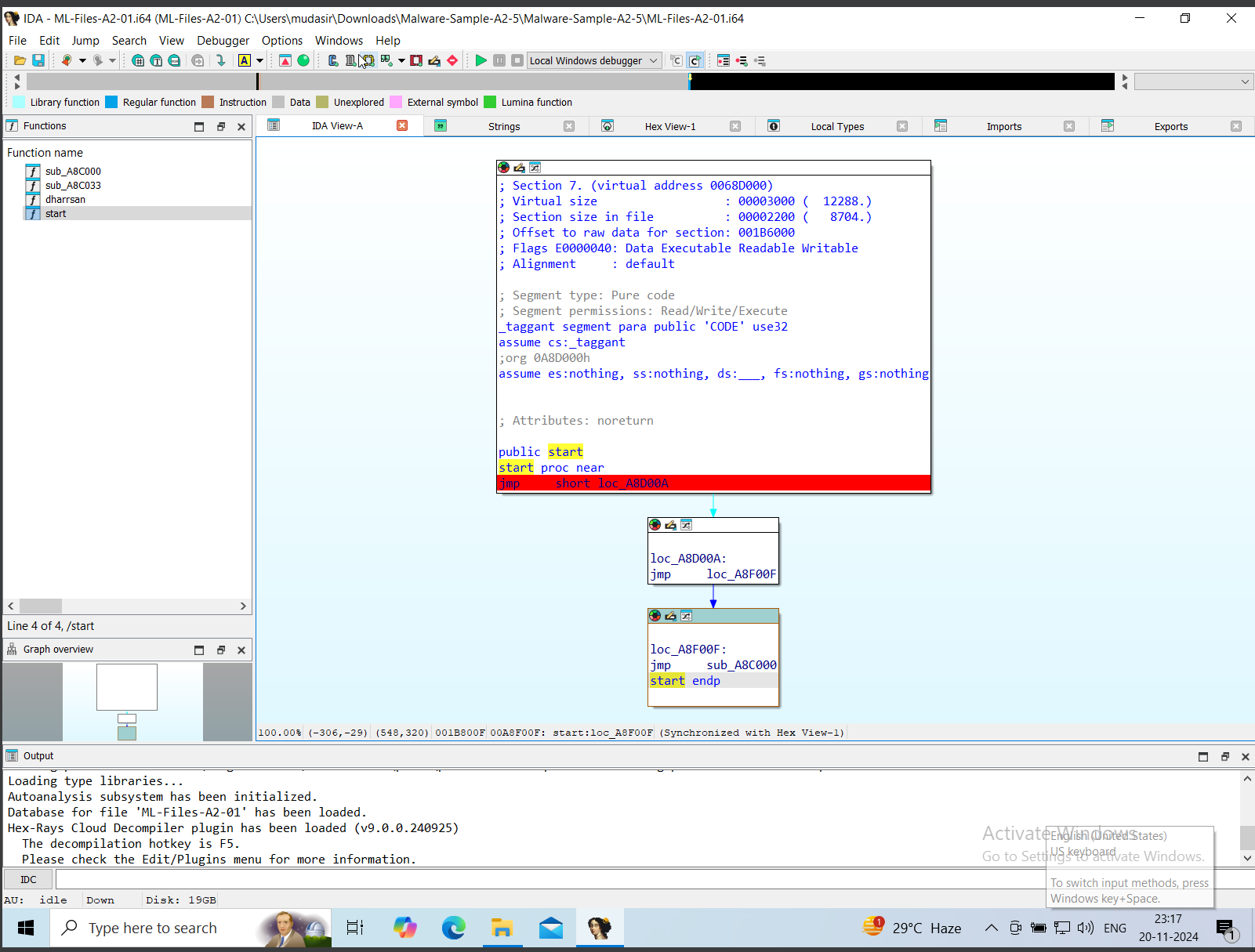
#### a) Create a Flowchart of the Main Function

1. Press F12 on the main function to view its graph.
2. Save the graph as an image using **File > Export to File**.



#### b) Identify and Document All Conditional Branches

1. Look for **JMP**, **JZ**, **JNZ**, **CALL**, and **RET** instructions in the control flow.
2. Highlight branches in the **Graph View**.



#### c) List Any Loops Present in the Code

1. In **Graph View**, identify circular paths or back edges in loops.
2. Look for patterns like:
   1. **x86**: cmp ...; jne/jz.

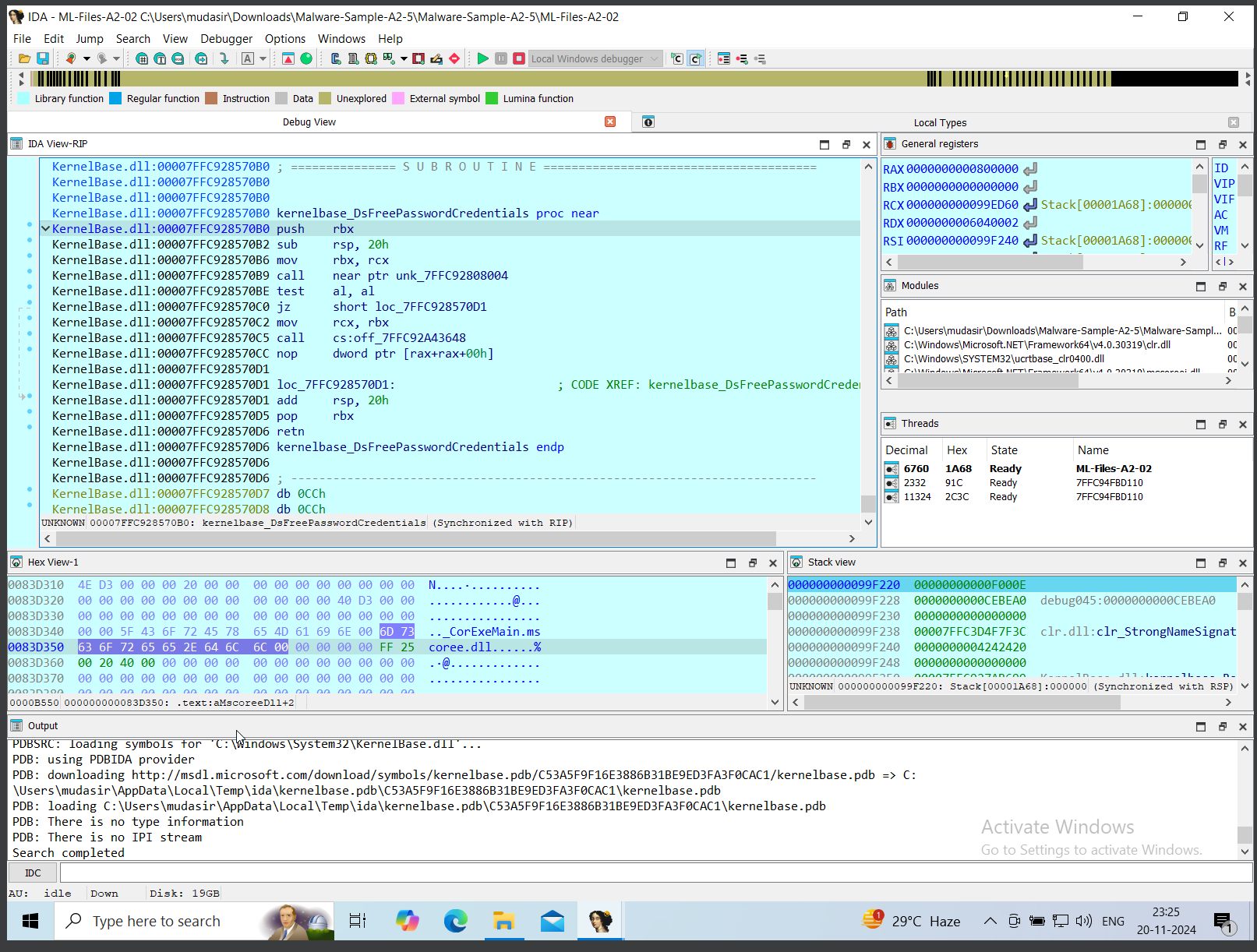




### ****4) Examine Cross-References****

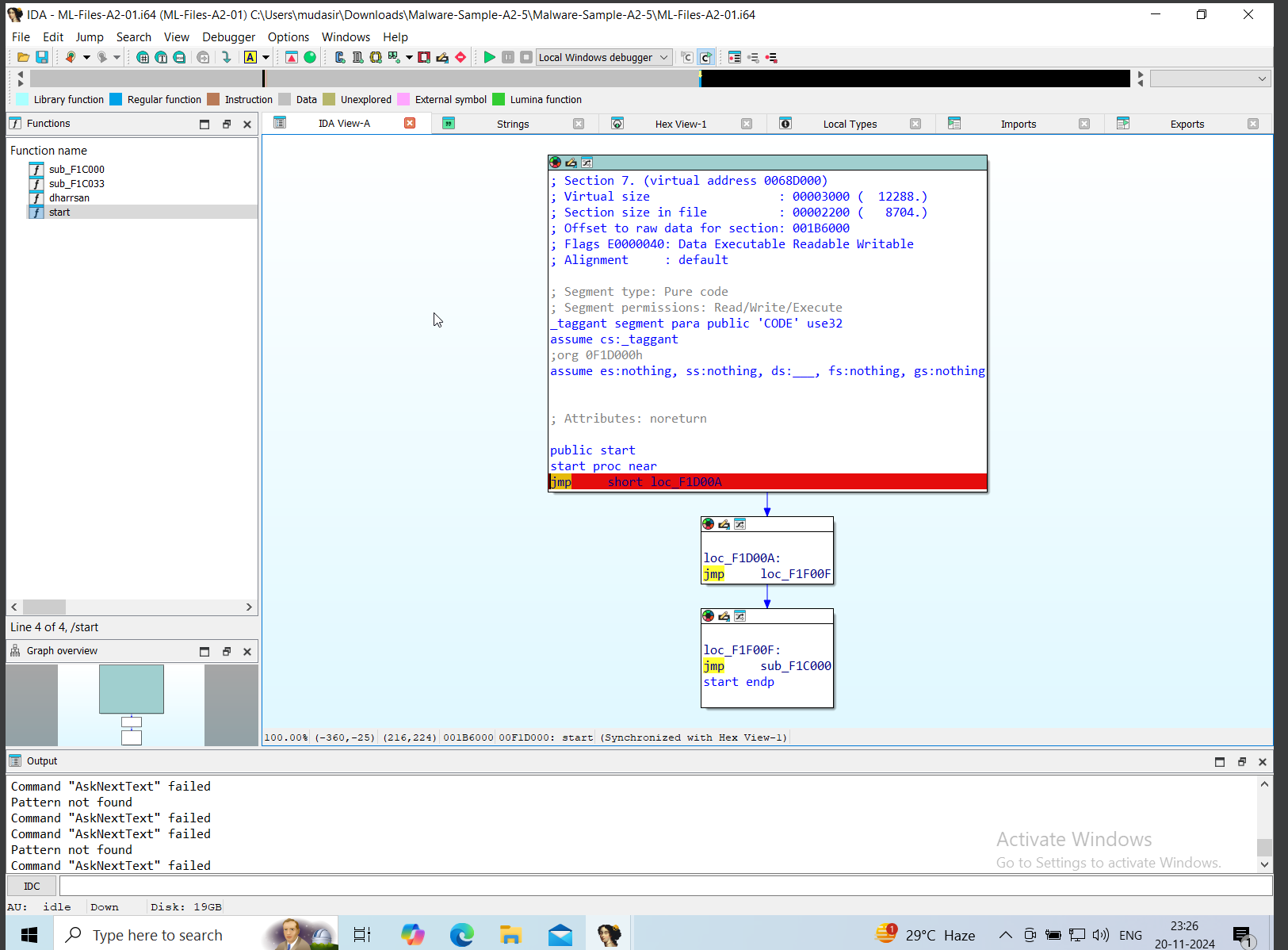
#### a) Find All References to the String "Password:"

1. In the **Strings Window**:
   1. Locate Password:.
   2. Double-click it and check **Xrefs** (cross-references).



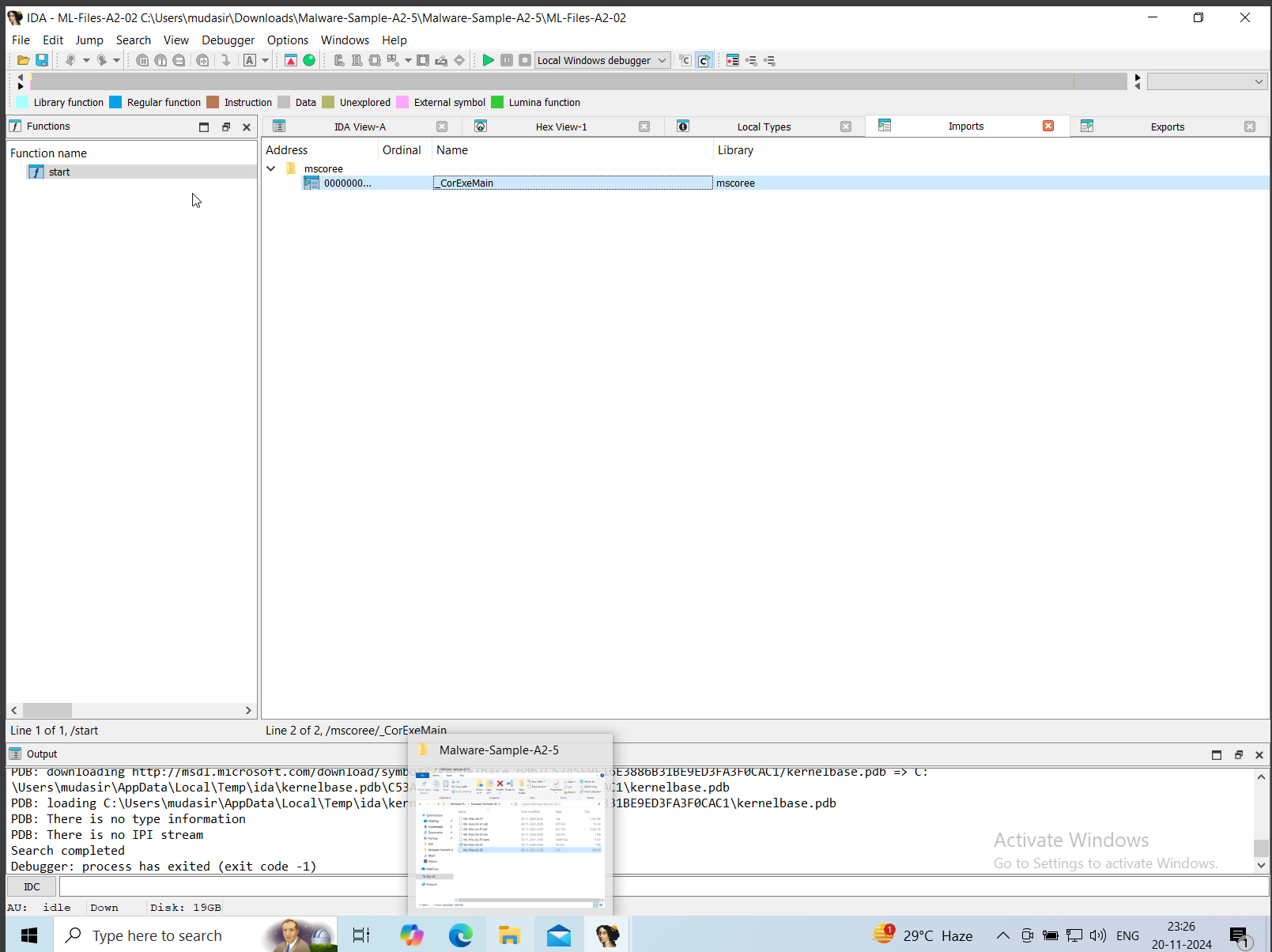
#### b) Document Functions Calling the Main Function

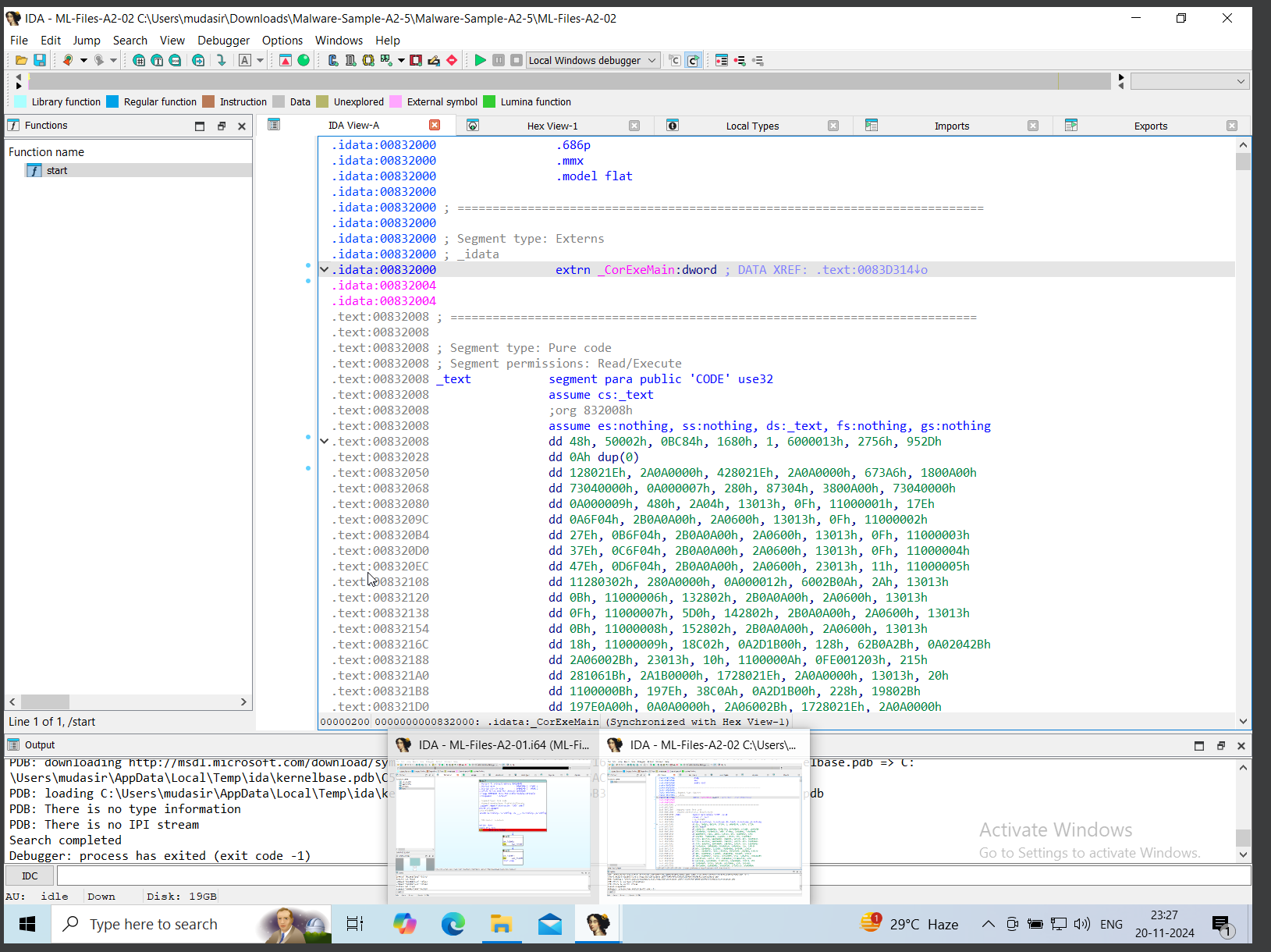
1. Open the **Functions Window** (Shift + F5).
2. Right-click main and select **"Jump to Xrefs to"**.



#### c) List External API Calls

1. In the **Imports Window**, note all external API calls.
2. Use **View > Open Subviews > Names** to cross-reference APIs.

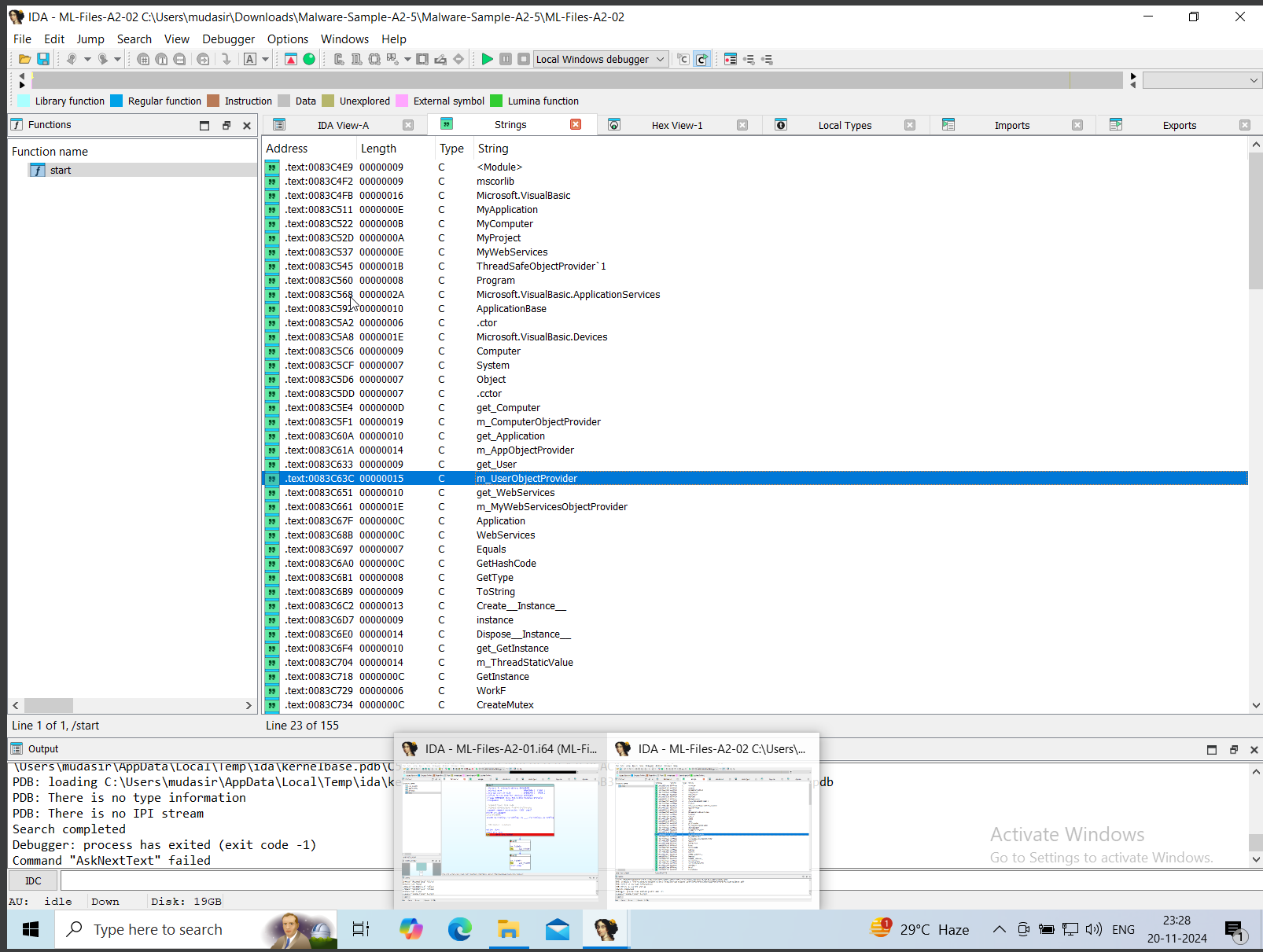




### ****5) Data Analysis Tasks****

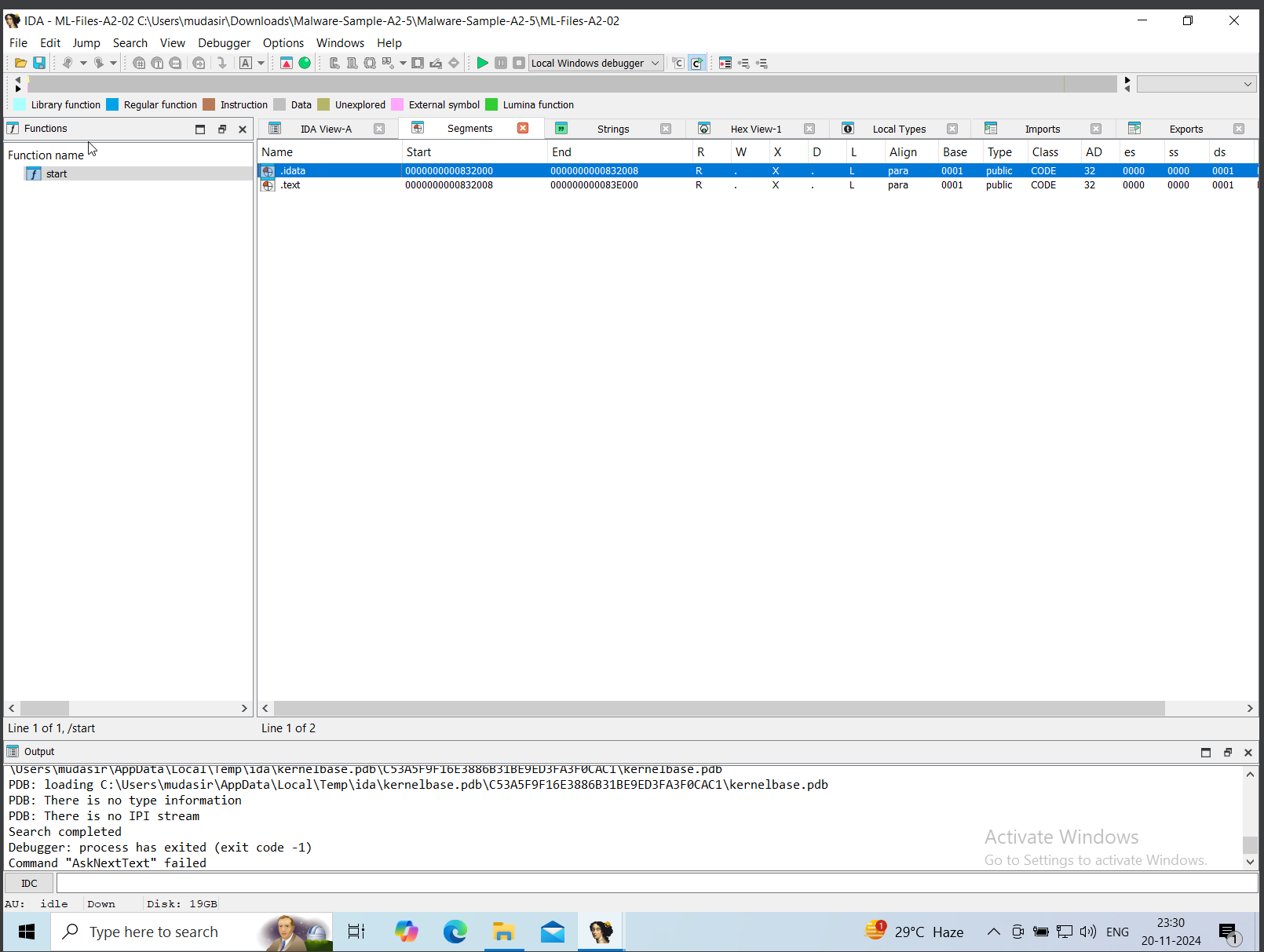
#### a) Locate and Document Any Encrypted Strings

1. Look for non-readable strings in the **Strings Window**.
2. Identify decryption routines:
   1. Follow functions processing the strings until they become readable.



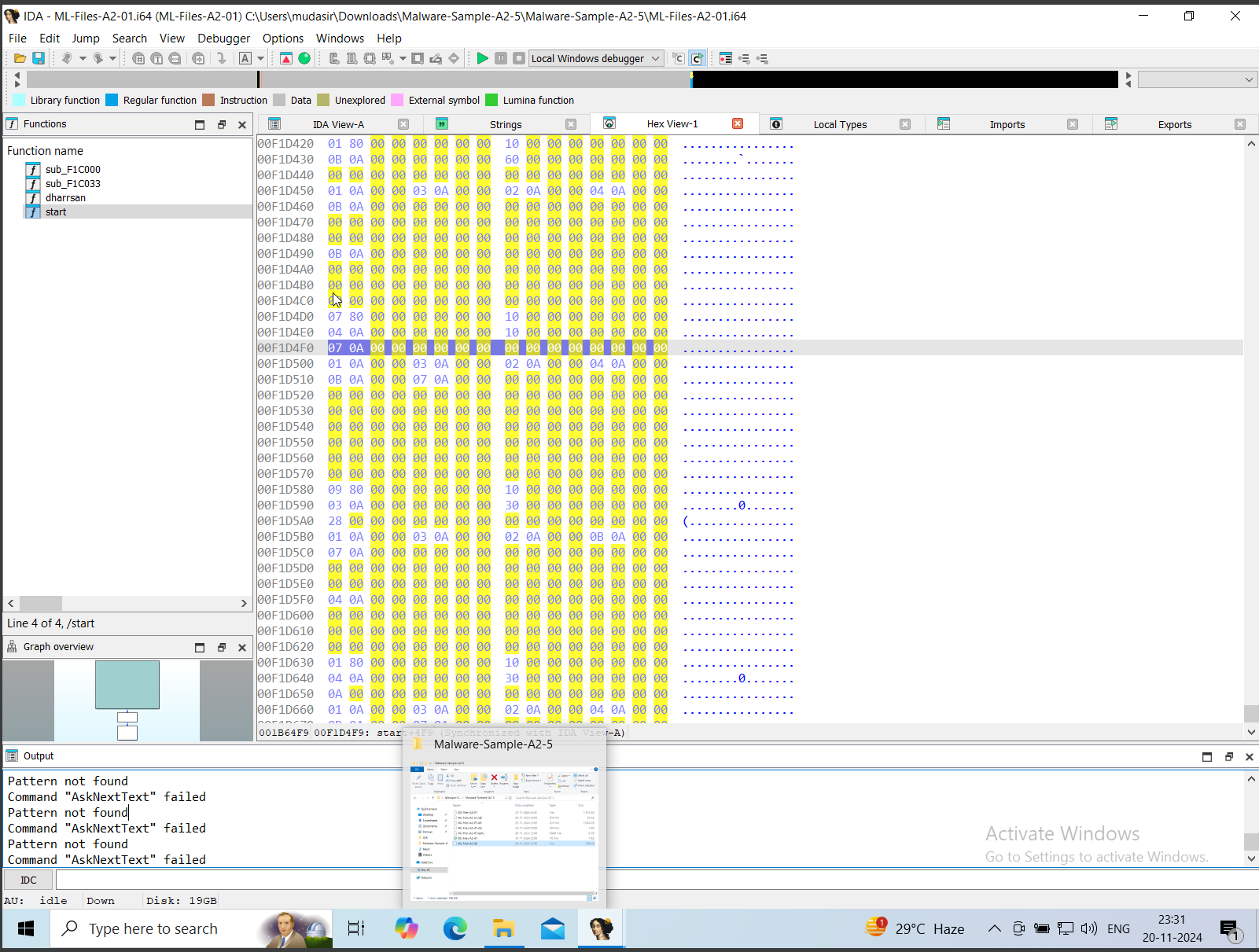
#### b) Identify Global Variables

1. Check **IDA View > Segments** for .data or .bss.
2. Document variables manipulated in these sections.



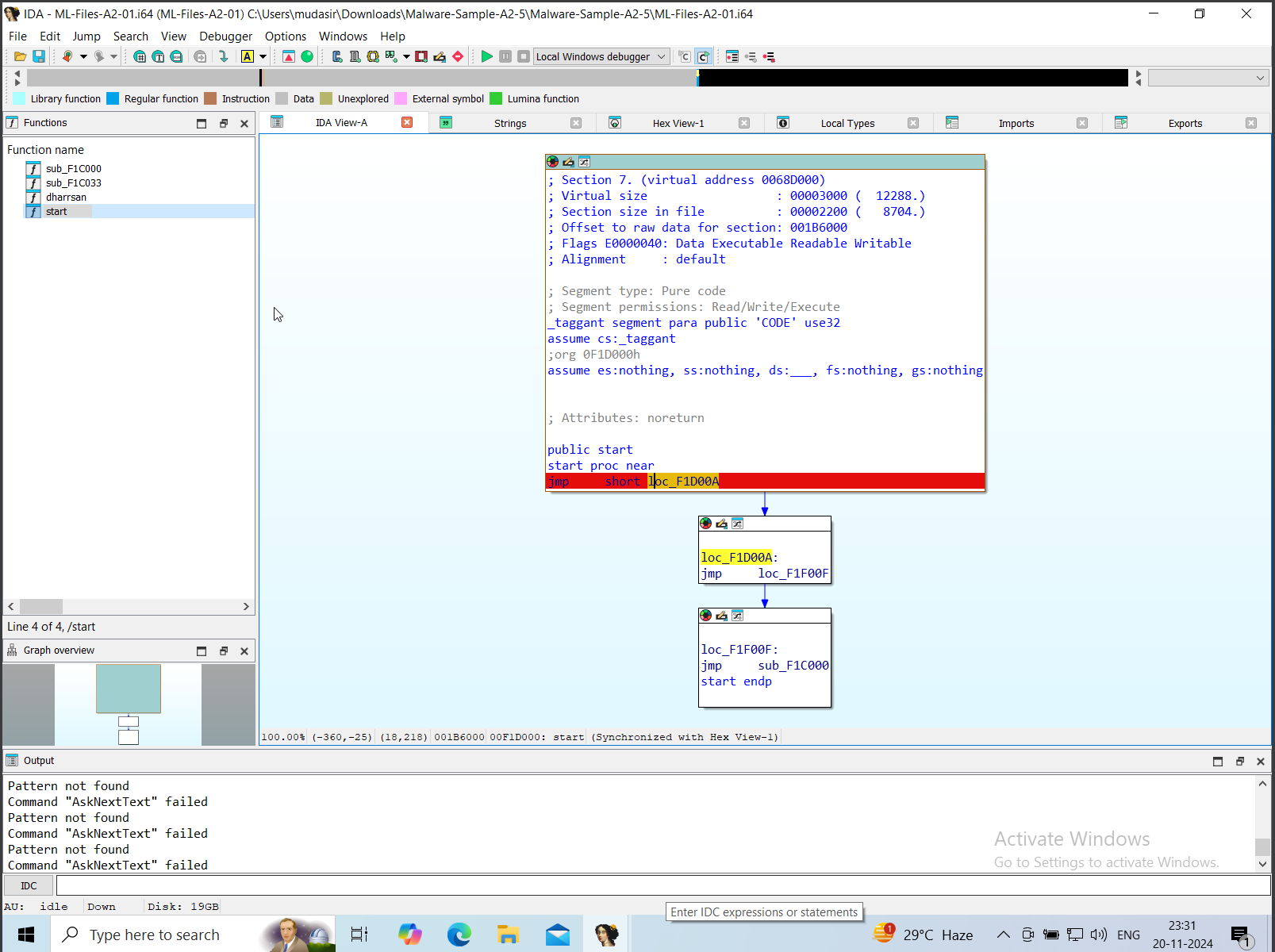
#### c) Document Interesting Data Structures

1. Look for arrays, structs, or buffers in the **Hex View** or the **Disassembly View**.



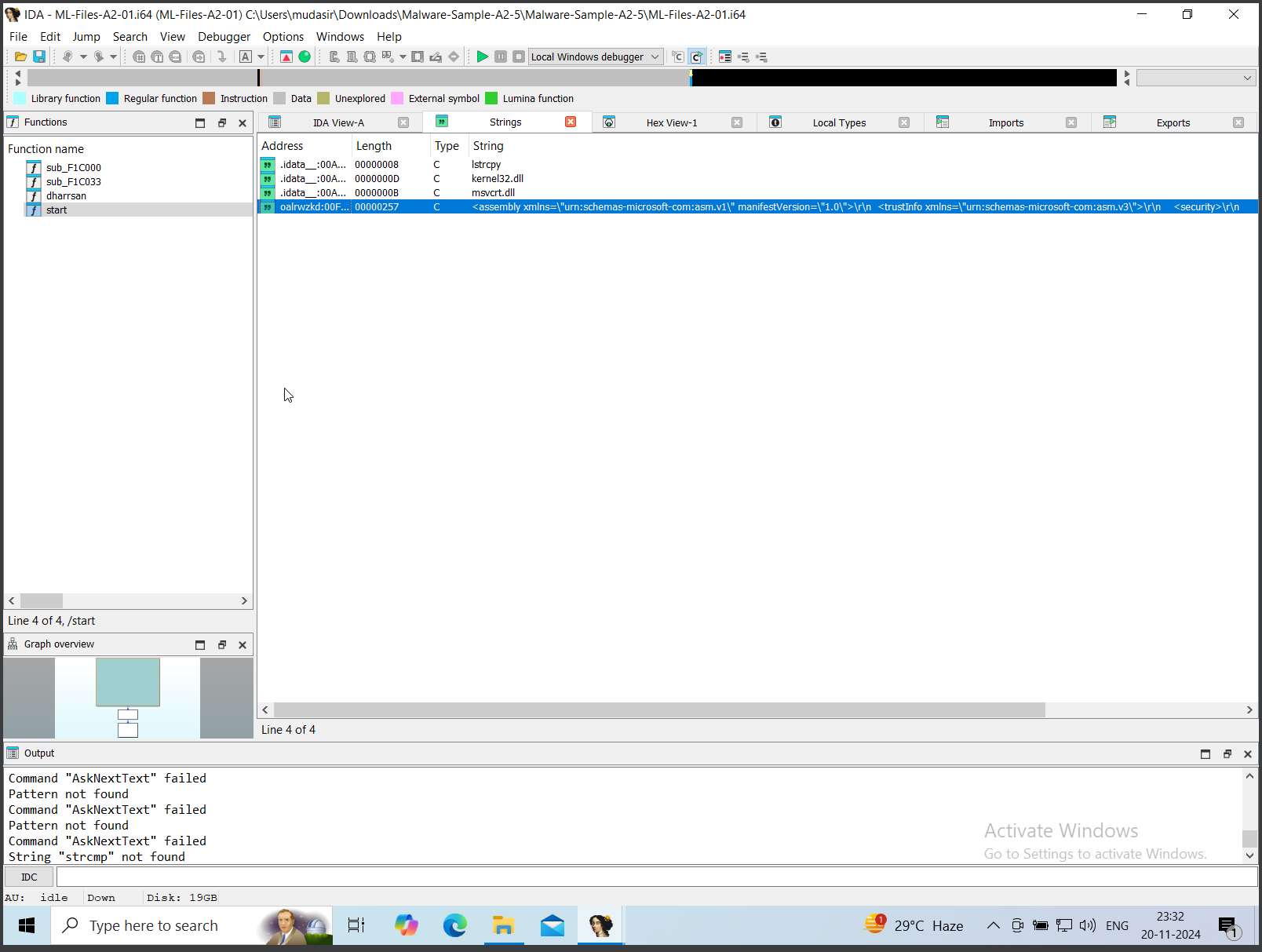
#### d) Document the Program’s Basic Functionality

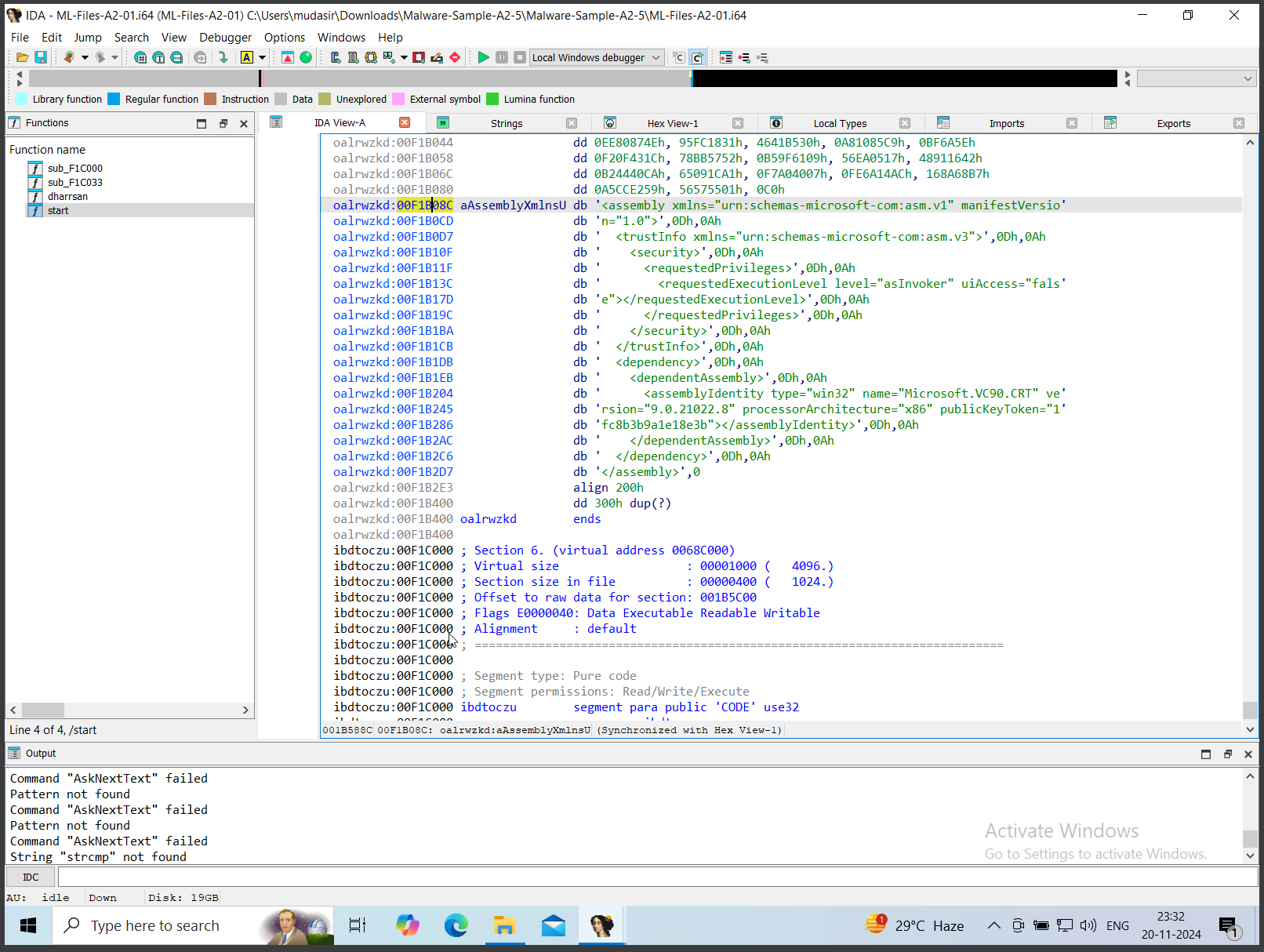
1. Trace from the main function to other key routines.
2. Use **comments** (;) to annotate findings.



#### e) Identify Potential Security Checks

1. Look for conditional checks involving:
   1. Strings (e.g., strcmp).
   2. Security-related APIs (e.g., CheckTokenMembership).

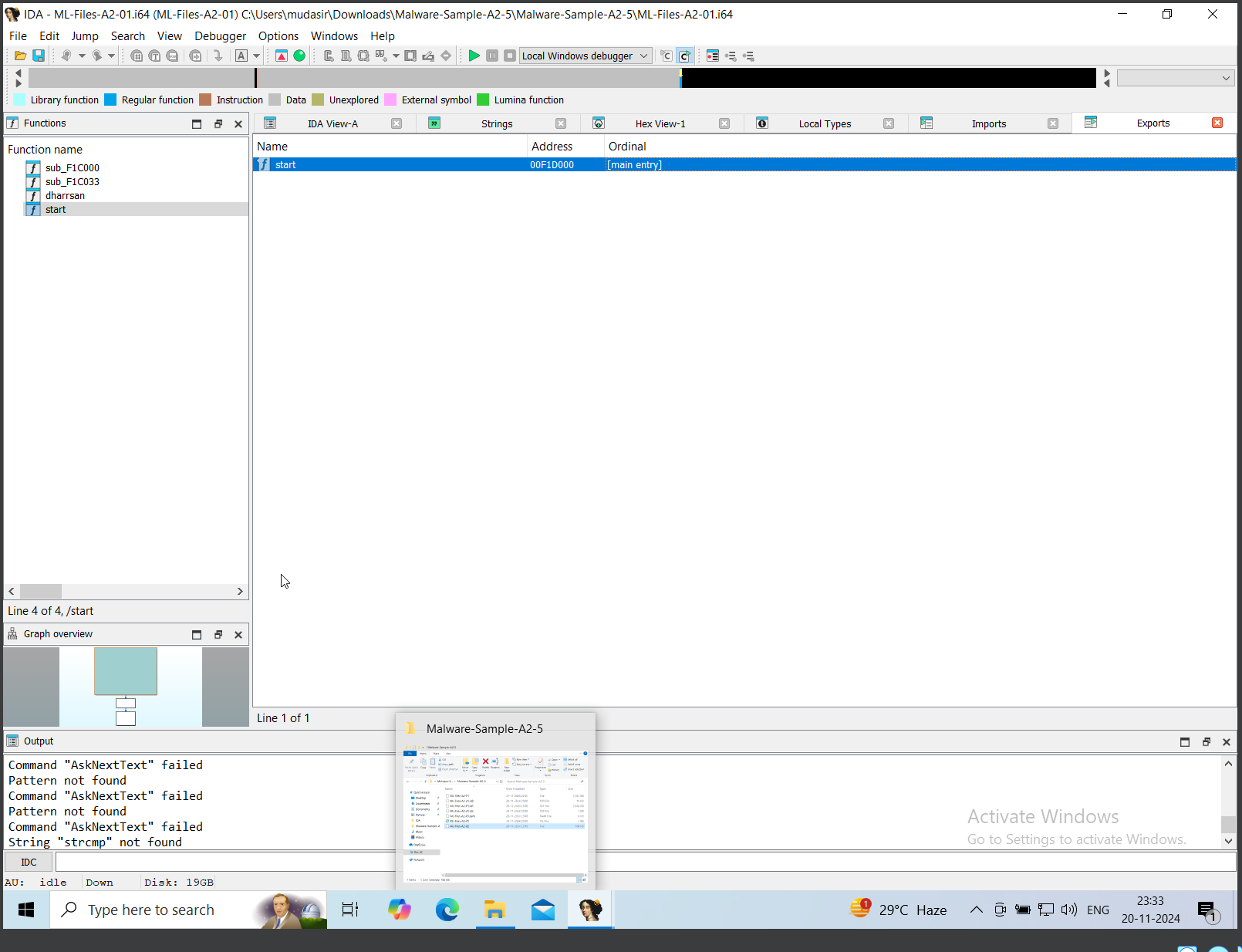


\

### ****6) Export and Document****

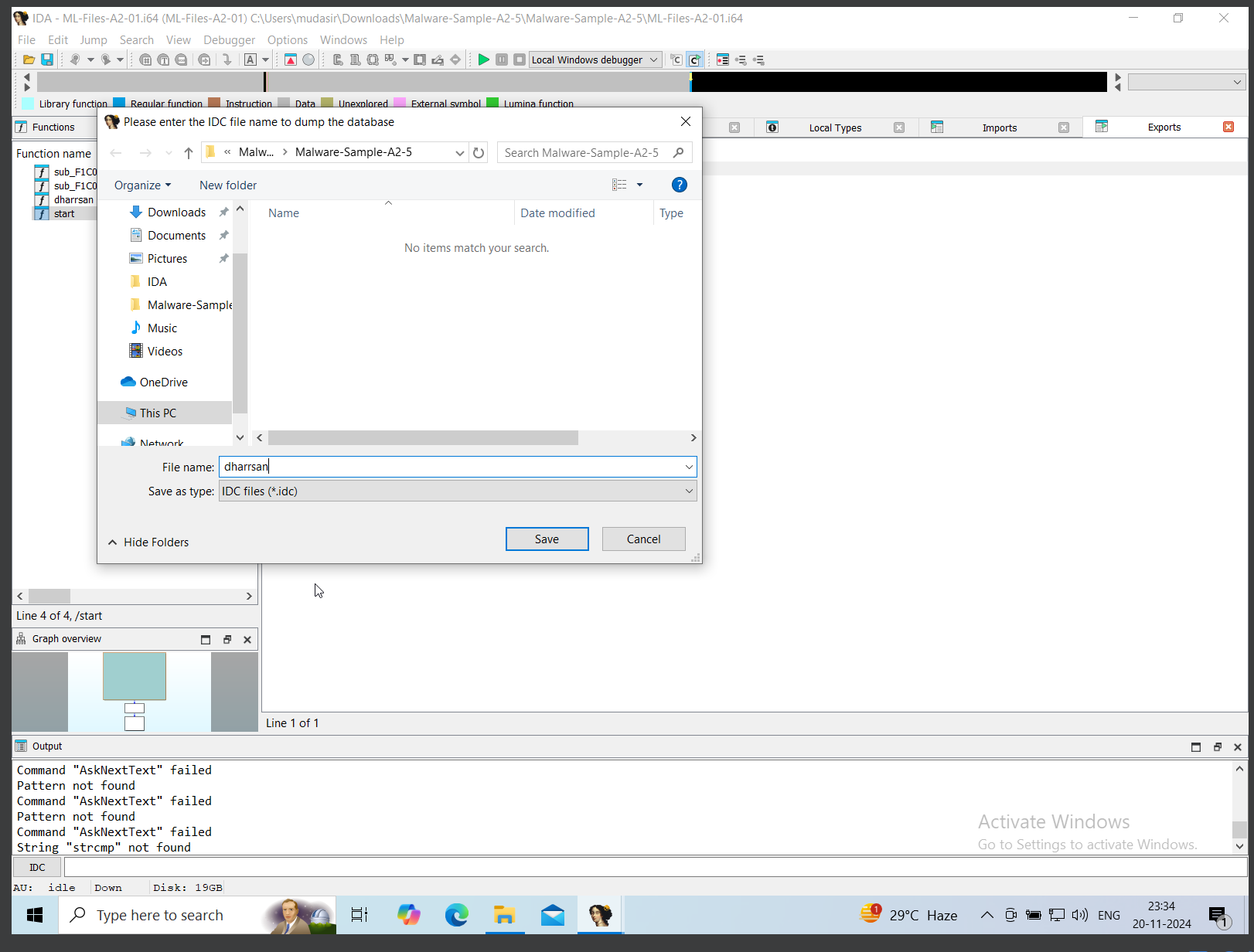
#### a) Generate an Exports Table

1. Navigate to **View > Open Subviews > Exports**.
2. Save it as a file with **Ctrl + S**.



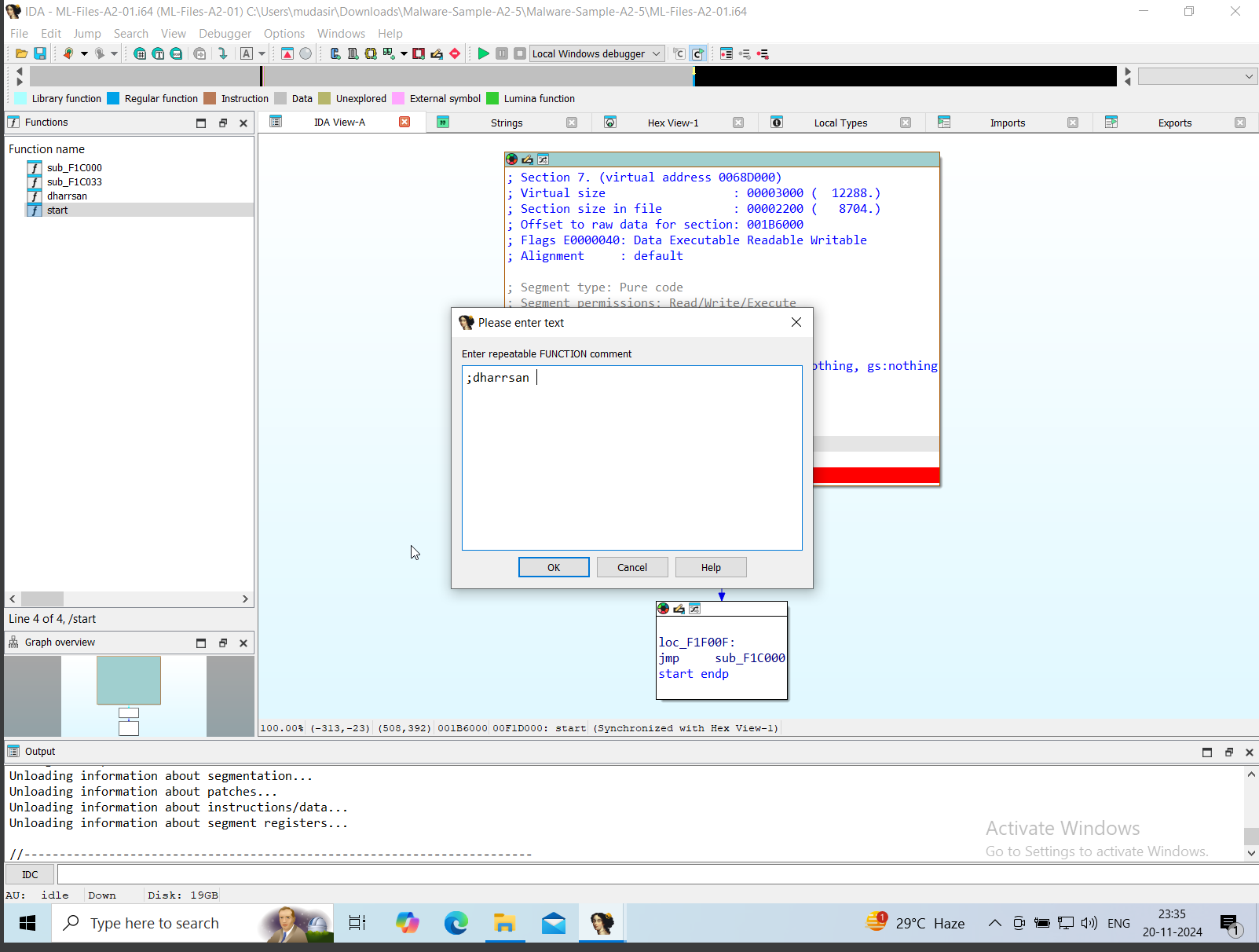
#### b) Save Your Renamed Functions

1. Use **File > Produce File > Create IDC File**.



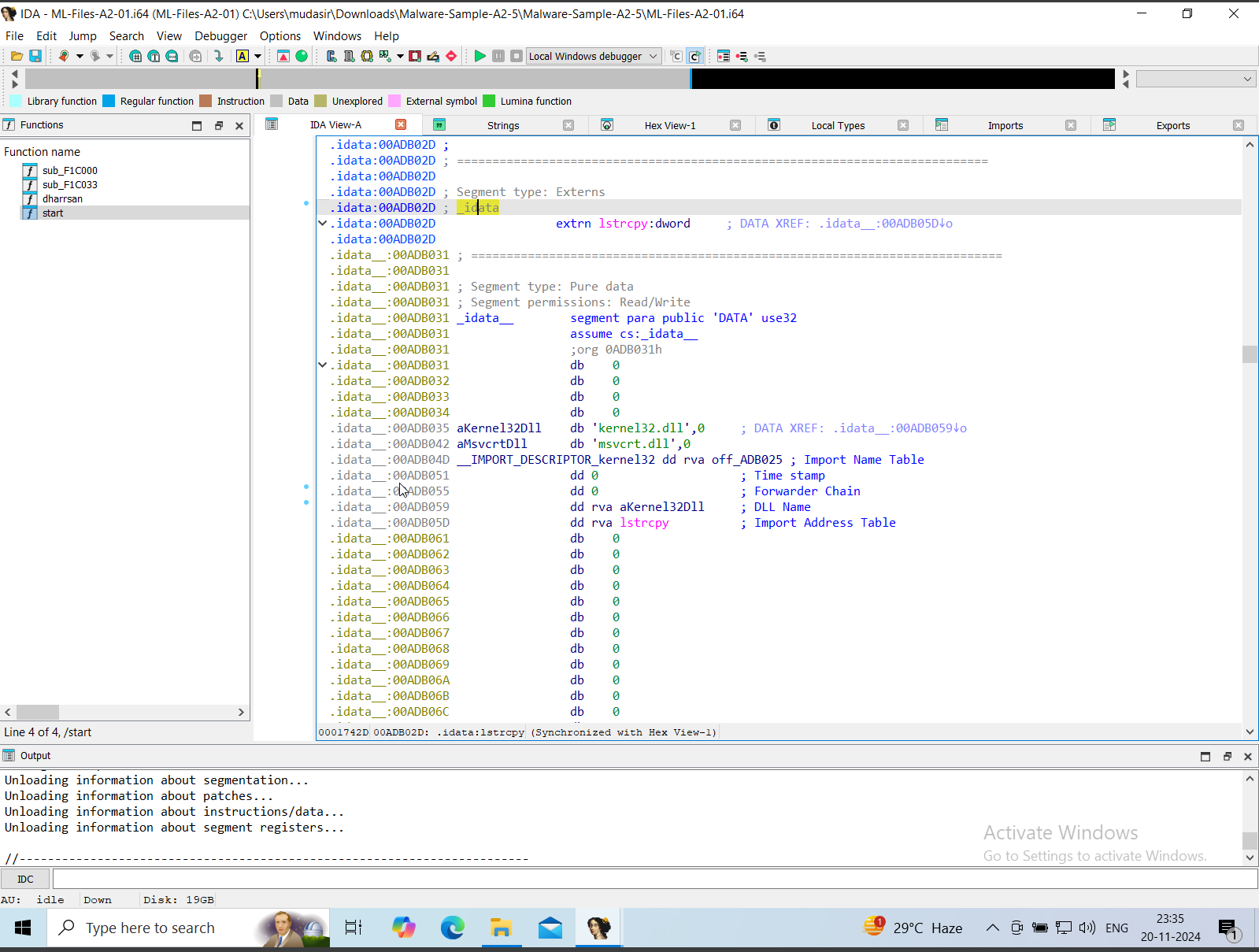
#### c) Create Function Comments for Key Procedures

1. Right-click a function and add comments (; or Ctrl + ;).



#### d) List Suspicious API Calls

1. Check the **Imports Window** for calls like:
   1. VirtualAlloc, VirtualProtect.
   2. Networking APIs (e.g., send, recv).



#### e) Document Any Network-Related Functions

1. Look for APIs like WSASocket, connect, or WinINet.



#### f) Identify File System Operations

1. Identify APIs such as CreateFile, ReadFile, and WriteFile.

