ANATOMY OF A SHELL CODE

Abhineet Ayan

Agenda:

1. Initialization: Finding own address in memory.

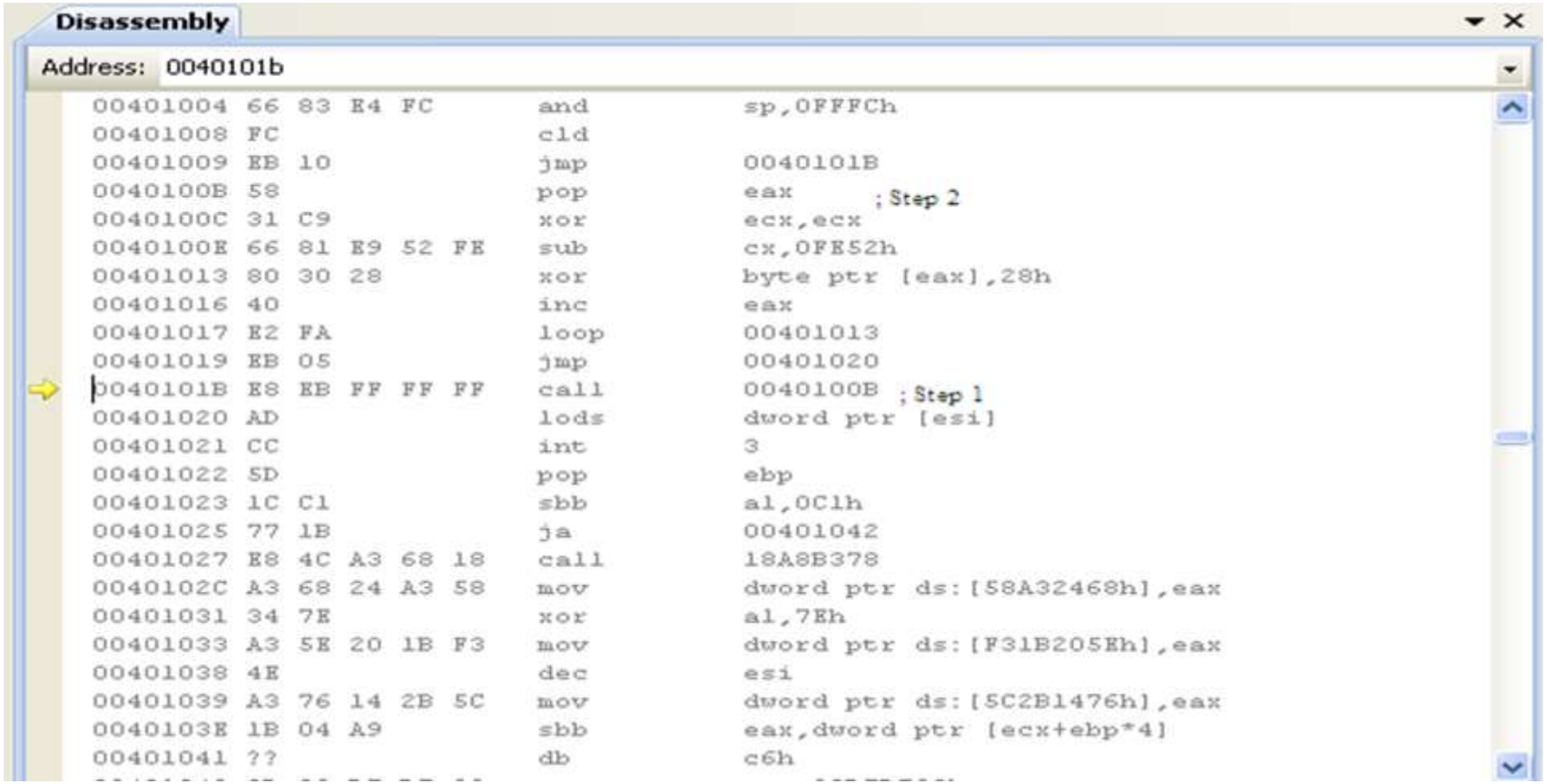
2. Decryption: Decryption of the code.

3. Loading Library Functions: Loading of desired functions.

4. Execution: Execution of the loaded functions to connect, download and execute the malicious file.

1. Initialization

- GetPC aka “Get Program Counter”.

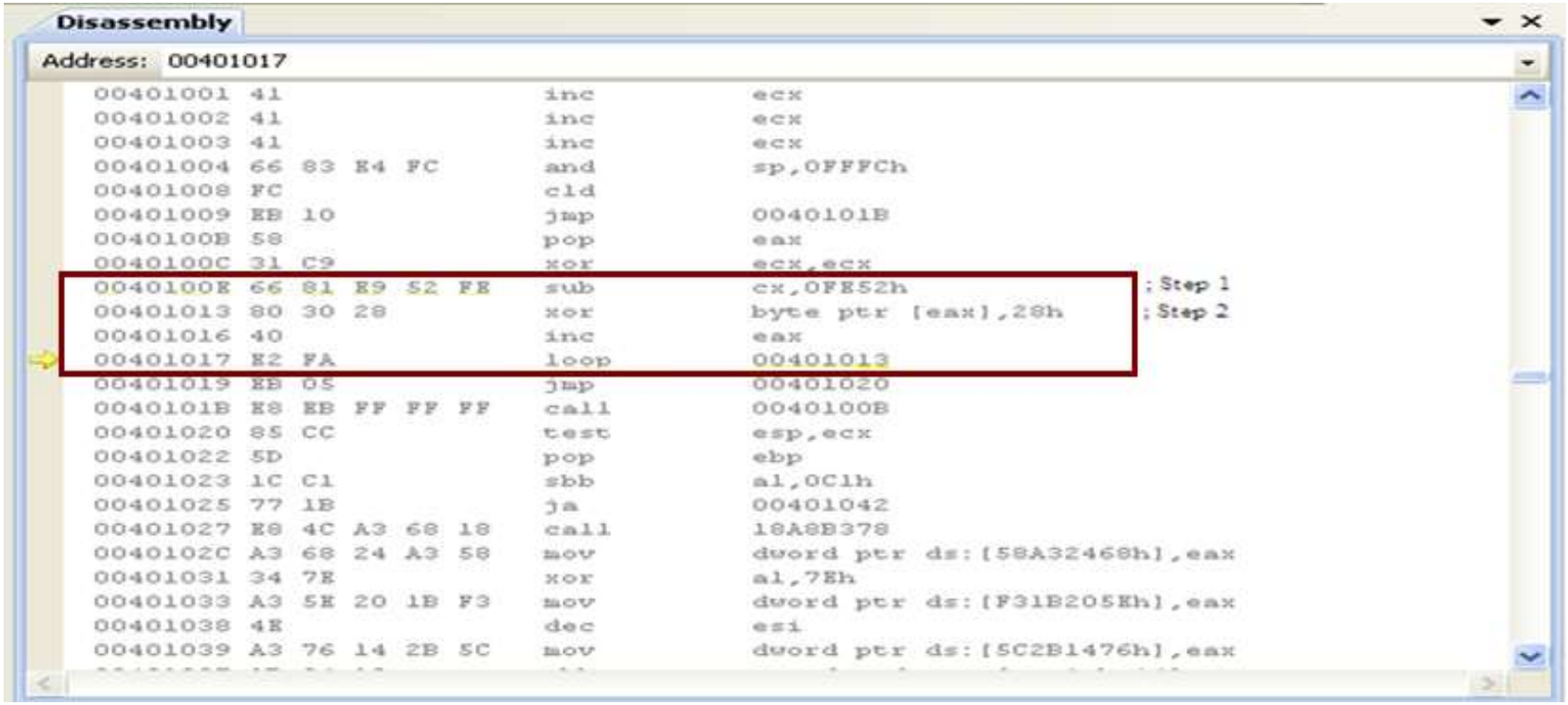


- Step 1: CALL instruction will push the offset from EIP and will jump to the specified line.
- Step 2: POP instruction will pop the offset pushed and will store it to the EAX.

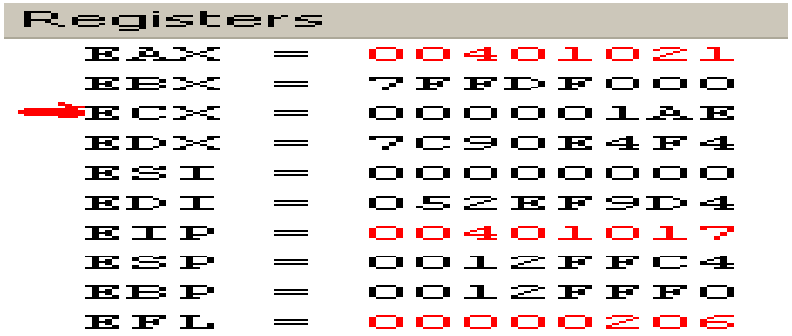
So, now the address is known and saved to EAX.

2. Decryption

- Decryption Loop:
  - Now after knowing its own address, it will decrypt the whole code to carry out the execution. The code before decryption is just garbage and will not accomplish anything. Let's have a look at Fig below:



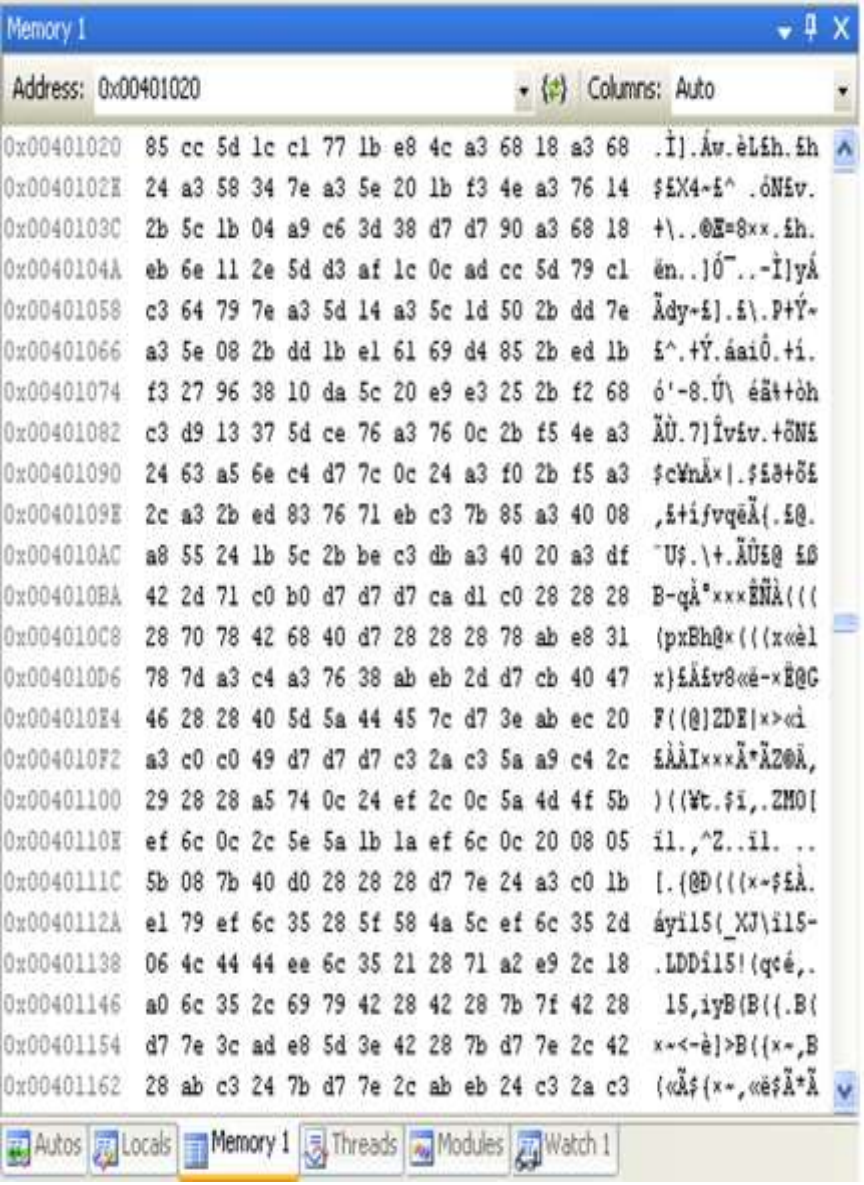
- Step 1: Set the ECX to number of bytes to decrypt. Below you can see the value of ECX.

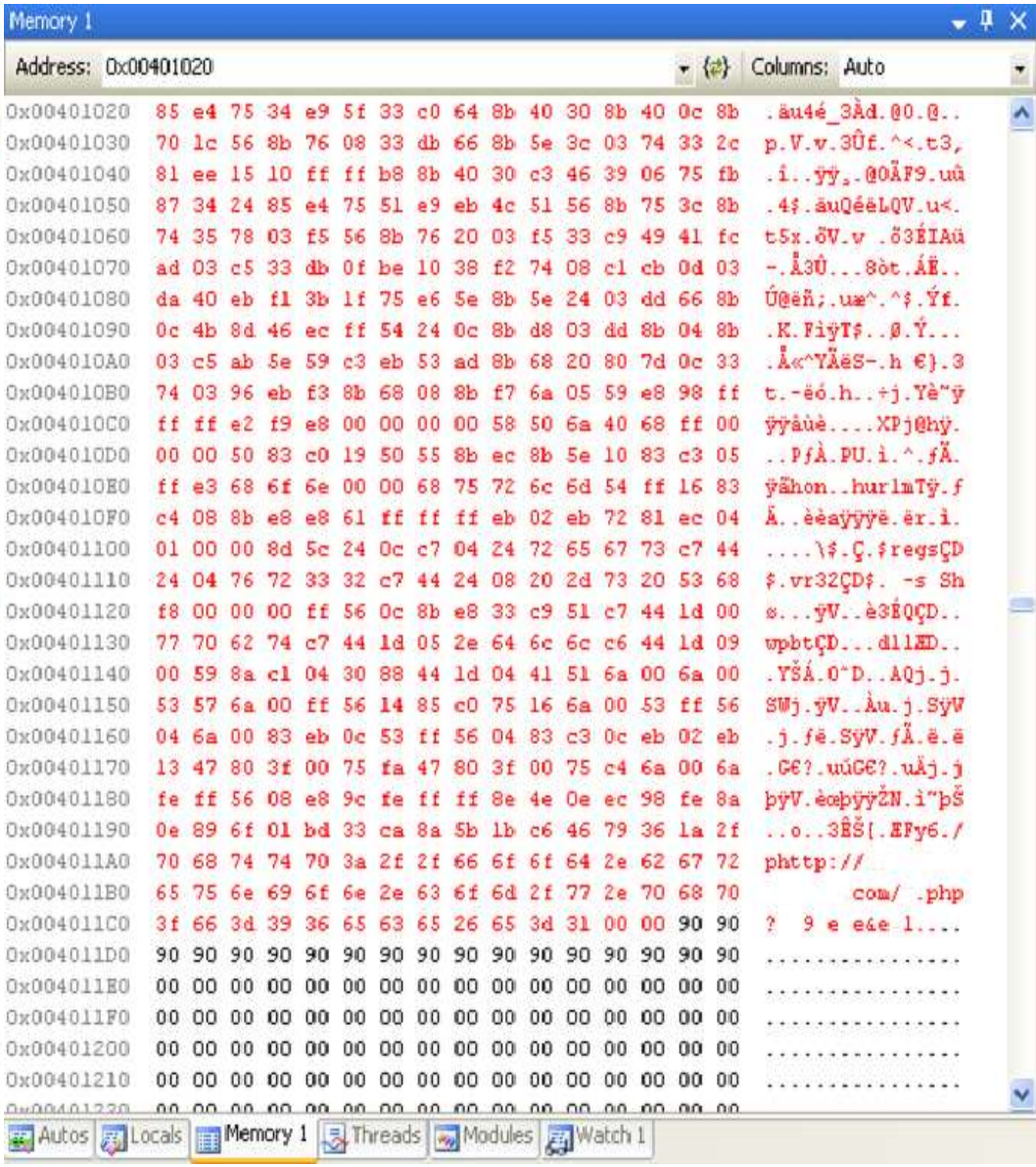


- Step 2: XOR each byte by 28h to yield actual data and keep incrementing EAX till ECX reaches 0 to exit the loop successfully.

The Decryption Loop keeps XORing each byte at the specified address with 28H to decrypt it. The loop will be executed the number of times equal to the Value represented in the ECX register.

Encrypted & Decrypted Memory

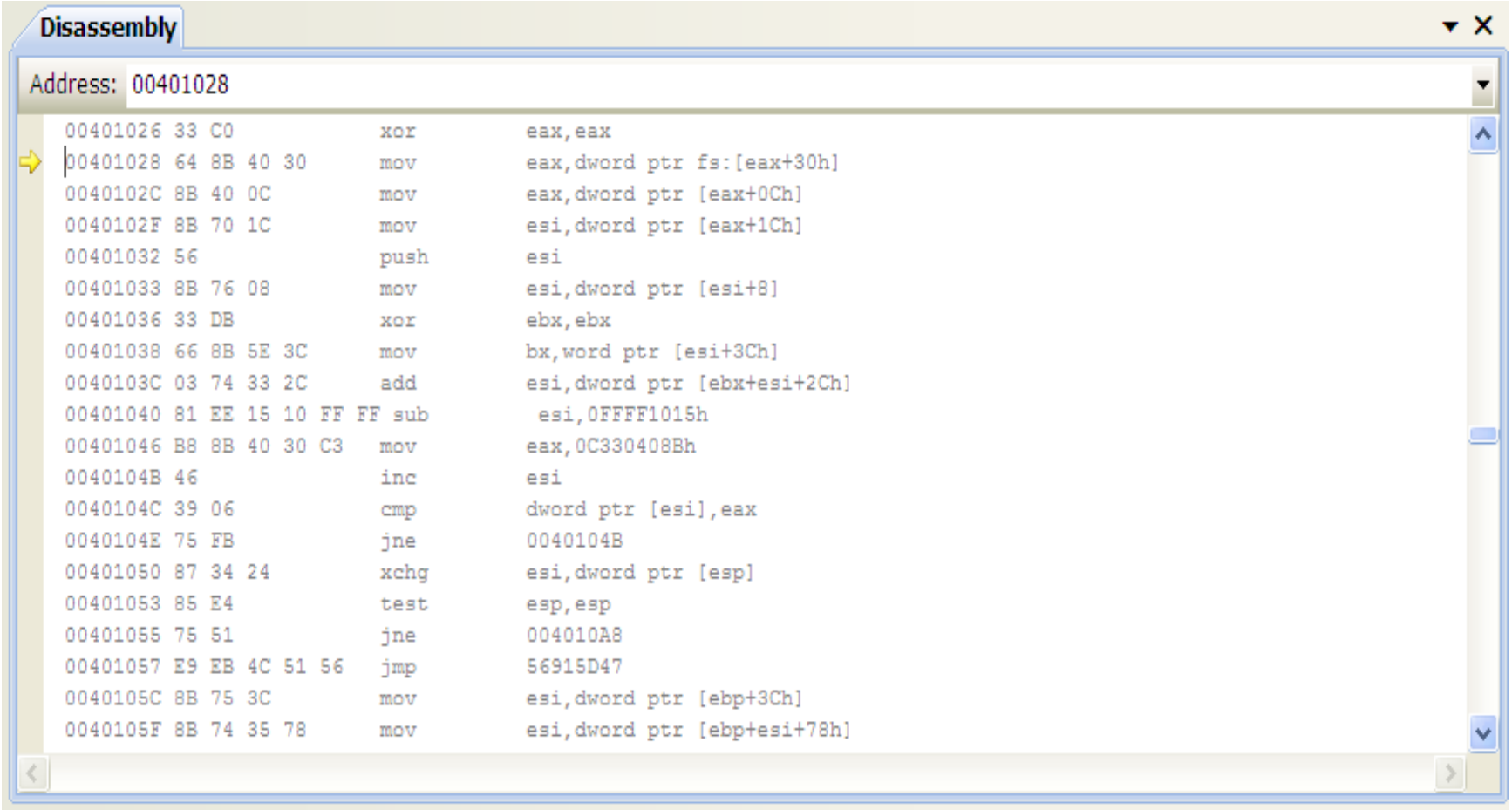




3. Loading Library Functions

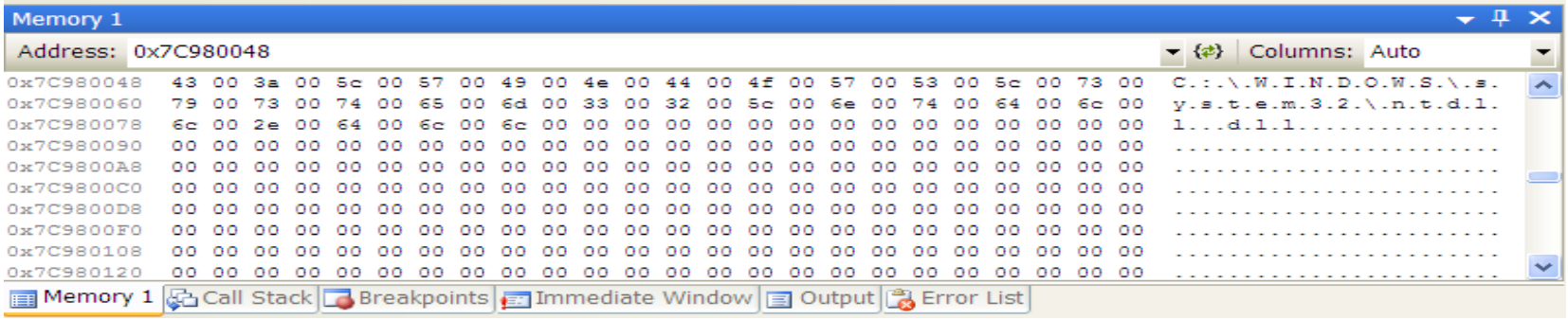
- PEB->Ldr (Process Environment Block Loader):
  - A common method to resolve the addresses of library functions needed is to get the base address of the kernel32.dll.
  - Addresses of desired functions by parsing the kernel32's PE-Header.
  - Process Environment Block (PEB) structure keep track of modules currently loaded in the processes' address space.
  - InInitializationOrder module list pointed to by the PEB->Ldr structure holds a linked list of modules.

So, shellcodes try to reach PEB->Ldr first and then navigates to the Kernel32.dll to retrieve the addresses of the desired functions.

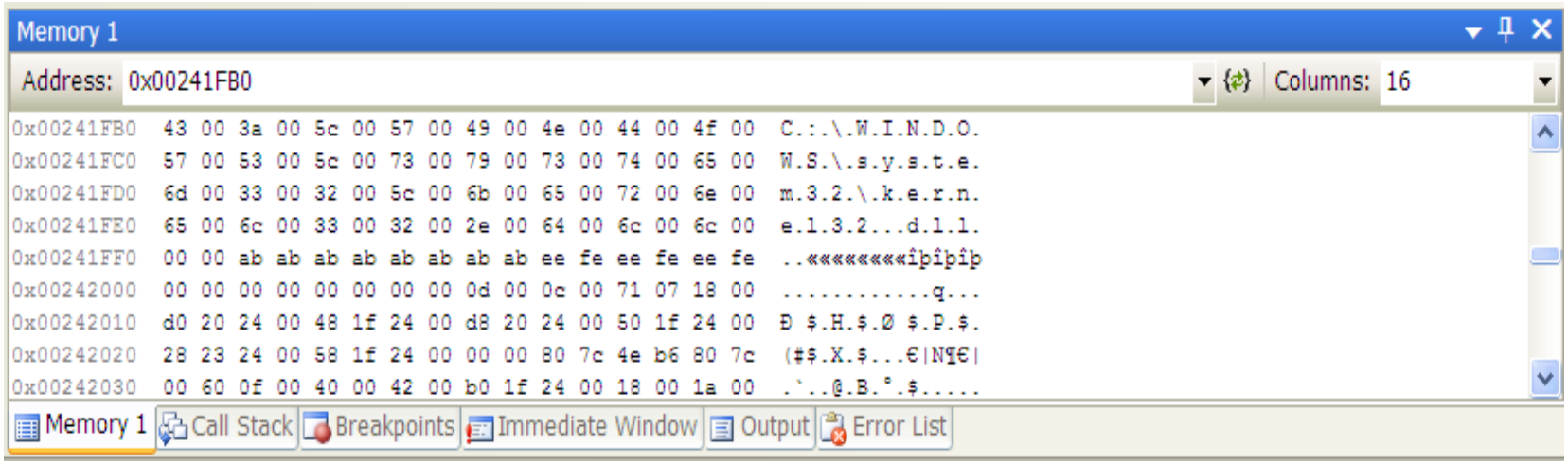


- Step 1: Get a pointer to the PEB.
- Step 2: Get PEB->Ldr.
- Step 3: Get PEB->Ldr.InInitializationOrderModuleList.Flink (1st Entry).

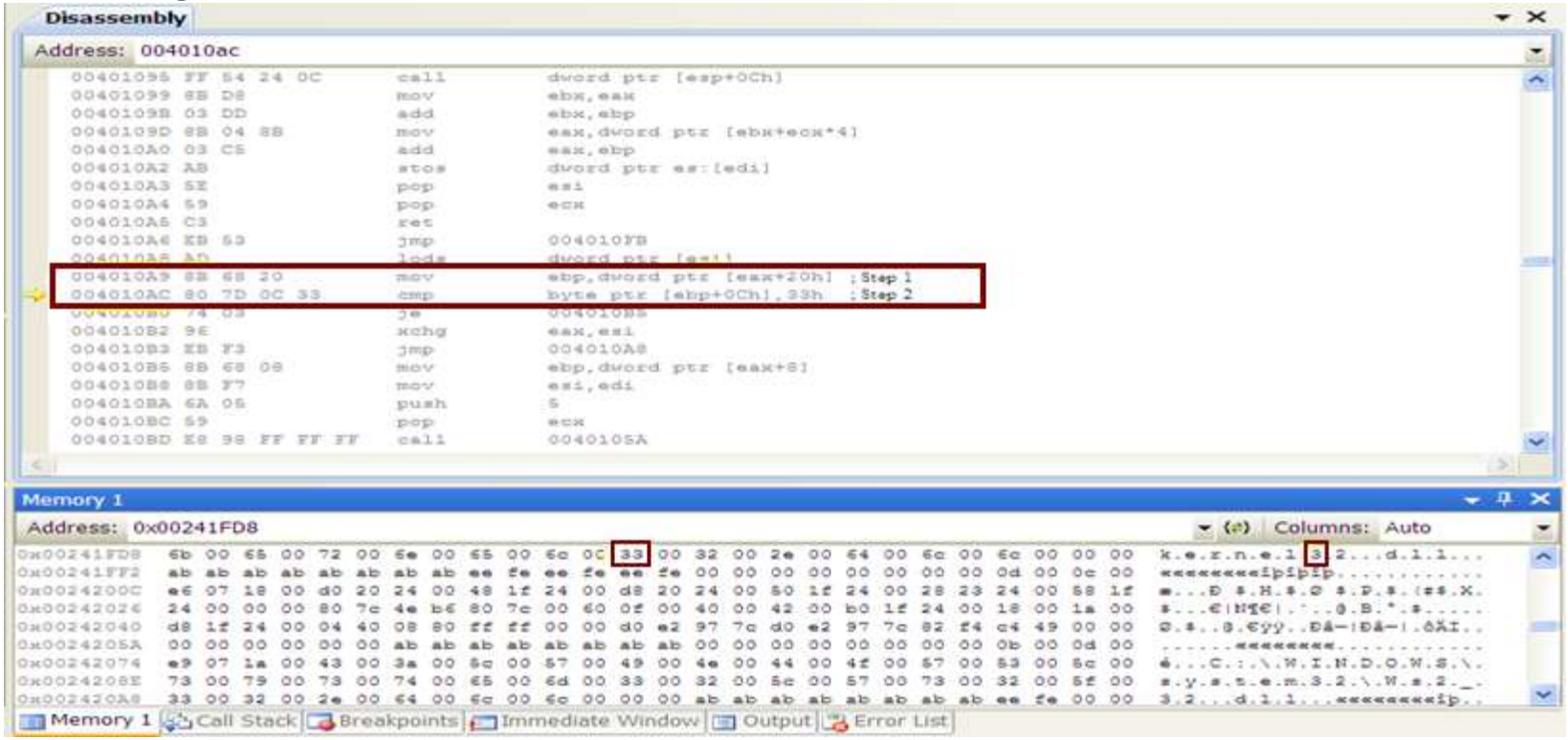
- The Memory shown below is after fetching the first entry of *PEB->Ldr.InInitializationOrderModuleList.Flink* and then “ntdll.dll” (1st Entry)



- Get PEB->Ldr.InInitializationOrderModuleList.Flink (2nd Entry):



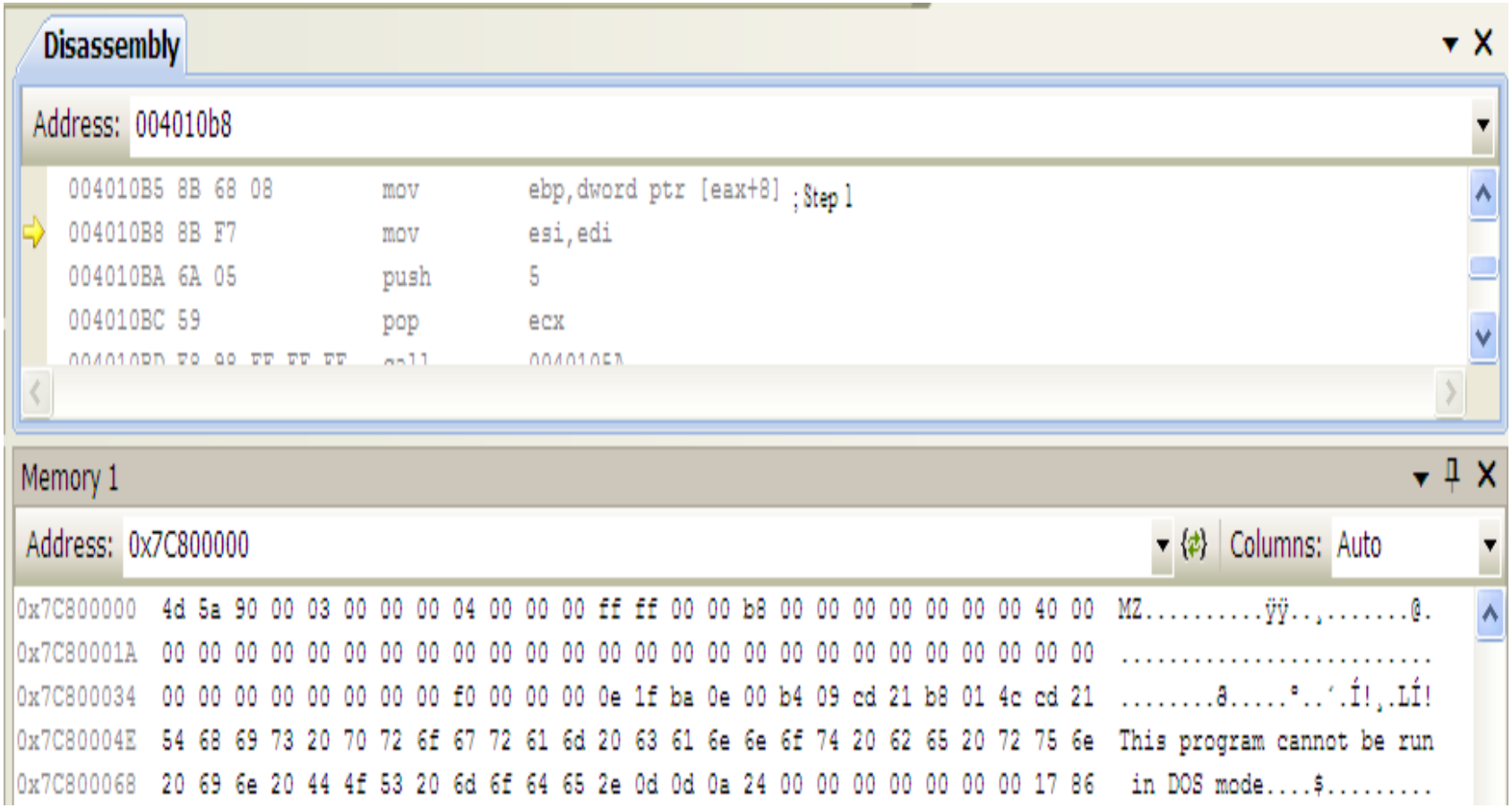
- Getting Address of Kernel32.dll:



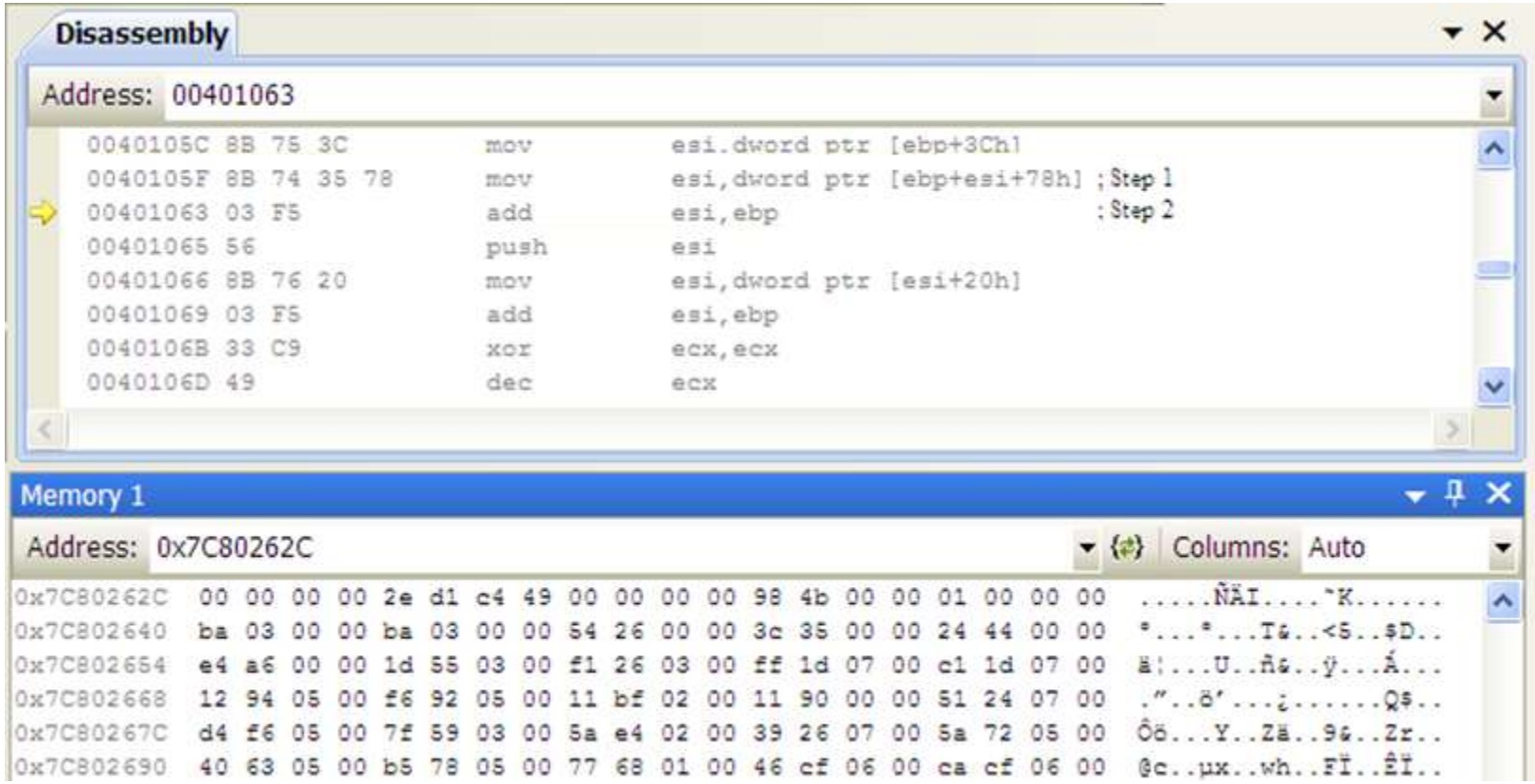
- Step 1: Move the address which shows the current DLLs' name to EBP.
- Step 2: Checks for the 12th Byte if equal to digit 3.

The second step comparison is done to make sure if the retrieved DLL is Kernel32.dll.

- Getting Kernel32.dll PE-Header:
  - After confirming the name of Kernel32.dll, move the “Base Address” of Kernel32.dll to EBP. Memory shows the PE-Header of Kernel32.dll.



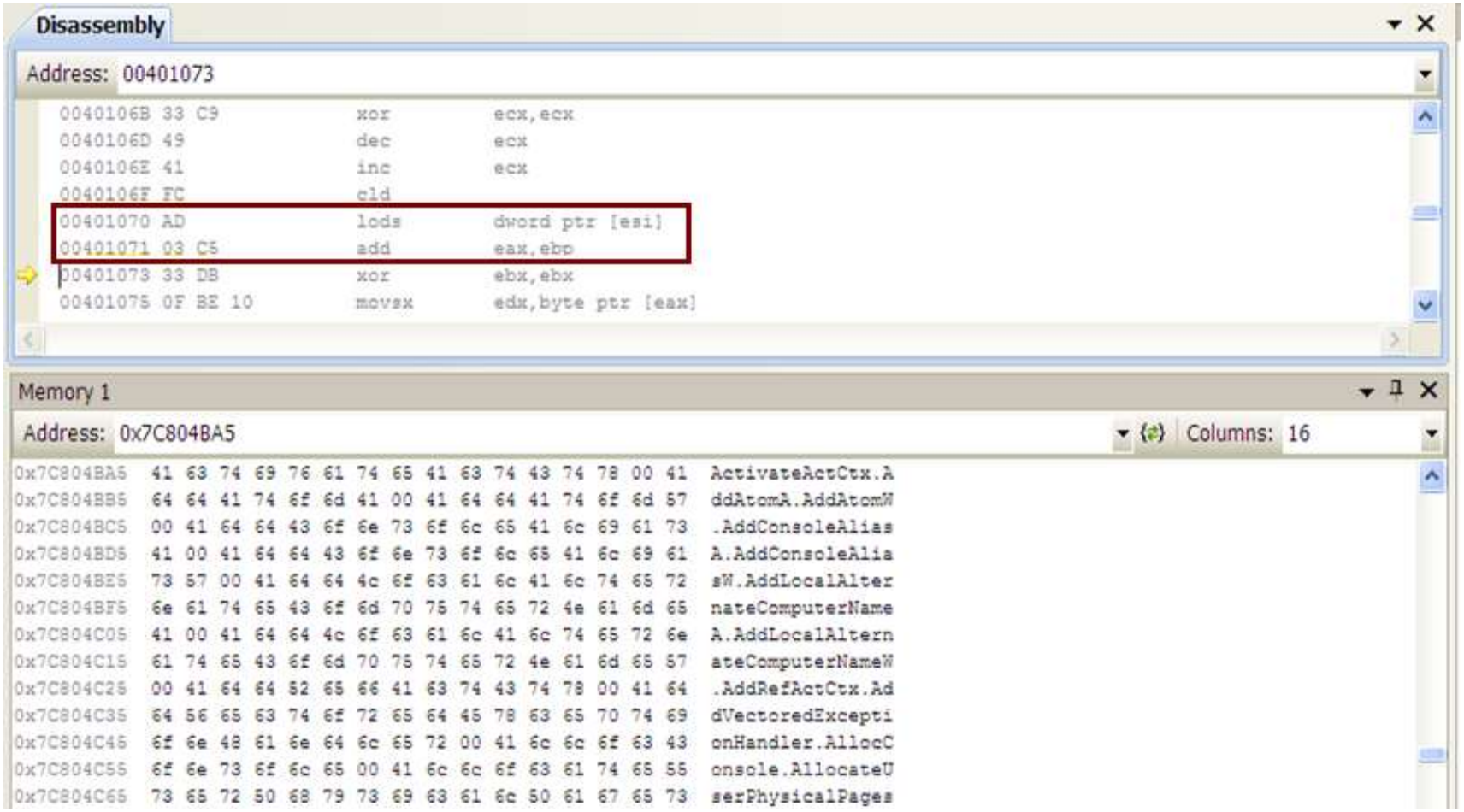
- Loading the Address of “IMAGE_DATA_DIRECTORY0” from PE-Header of Kernel32.dll:



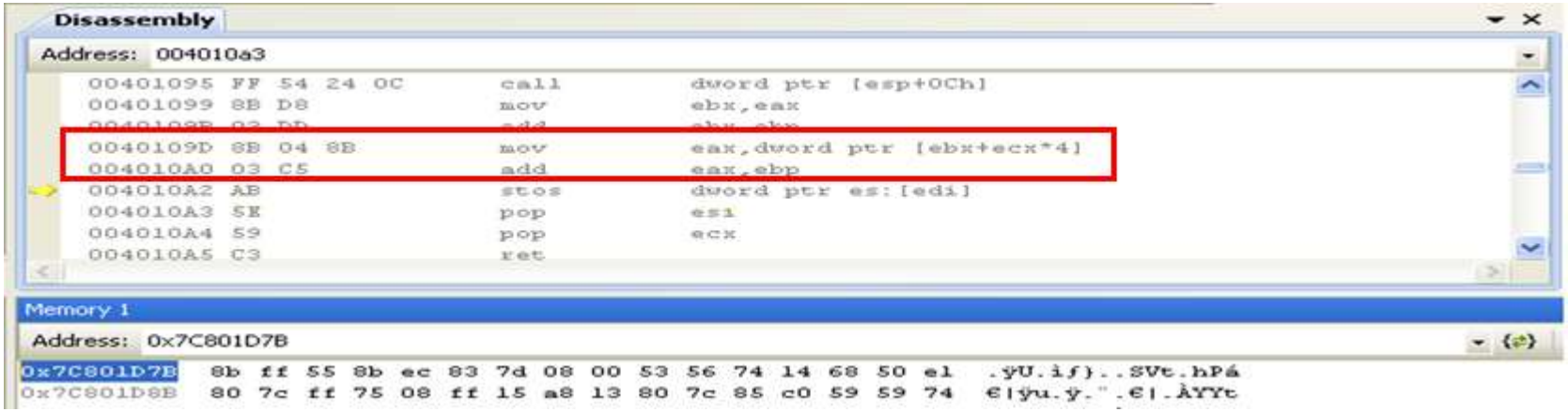
- Step 1: The offset for “IMAGE_DATA_DIRECTORY0” is loaded in ESI.
- Step 2: The actual address is loaded in ESI. (EBP is storing the base address of PE-Header).

The offset of “IMAGE_DATA_DIRECTORY0” is the RVA of “Export Directory”.

- Loading the Names of Functions of Kernel32.dll:
  - The rounded code will refer to the actual address of the member “AddressOfNames”.
  - The Memory window shows the loaded Names from Kernel32.dll.



- Resolving Address Of Desired Functions:

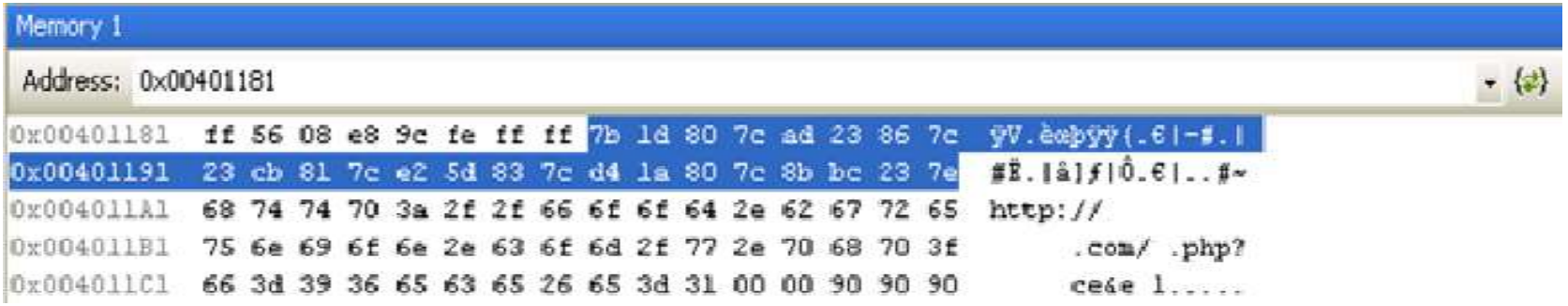


  - For eg. Resolving address of LoadLibraryA.

  - EAX contains address of LoadLibraryA.

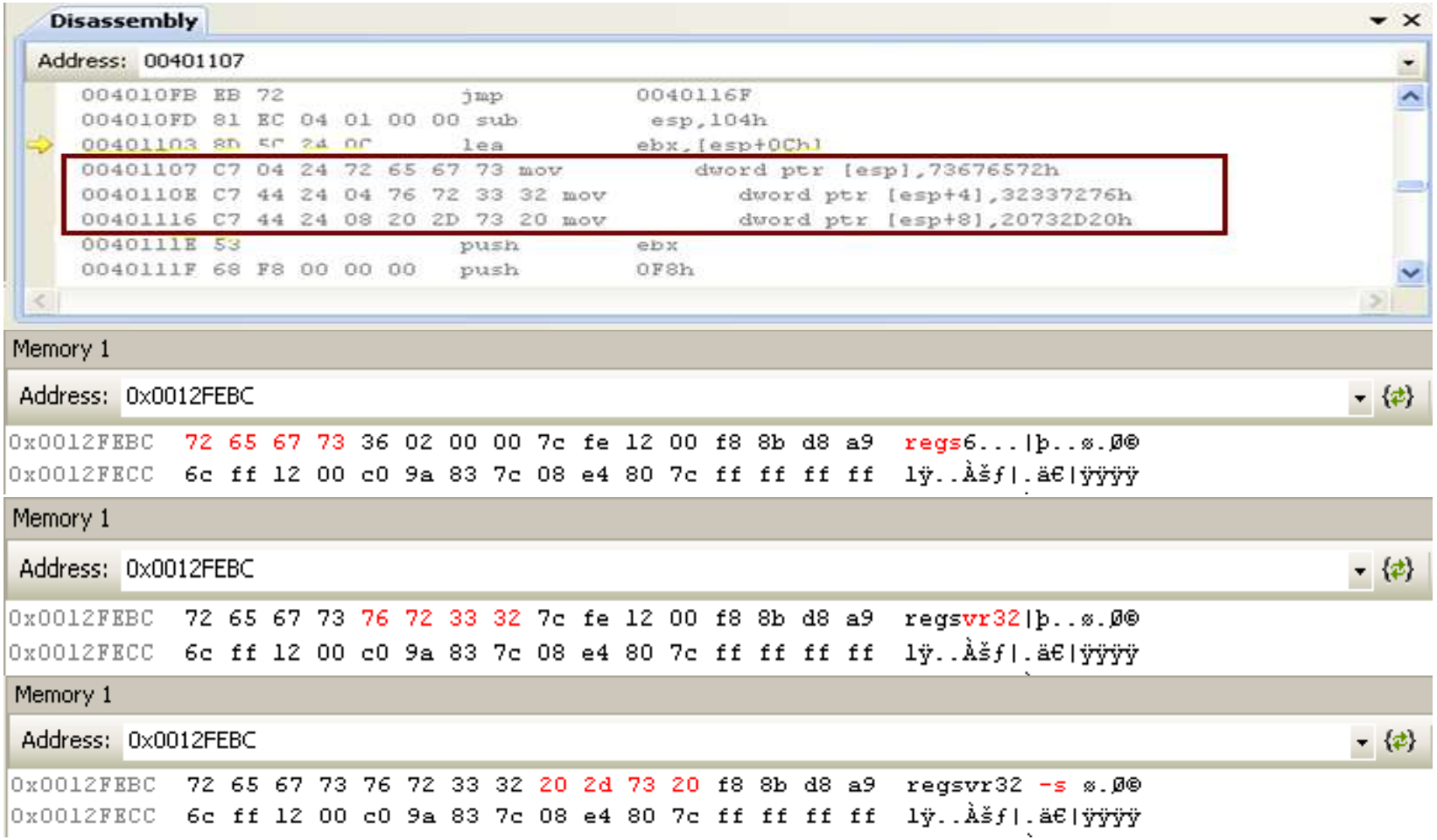
Totally five functions are resolved.

The Function Names are: *LoadLibraryA, WinExec, TerminateThread, GetTempPathA, VirtualProtect and UrlDownloadToFileA* respectively.

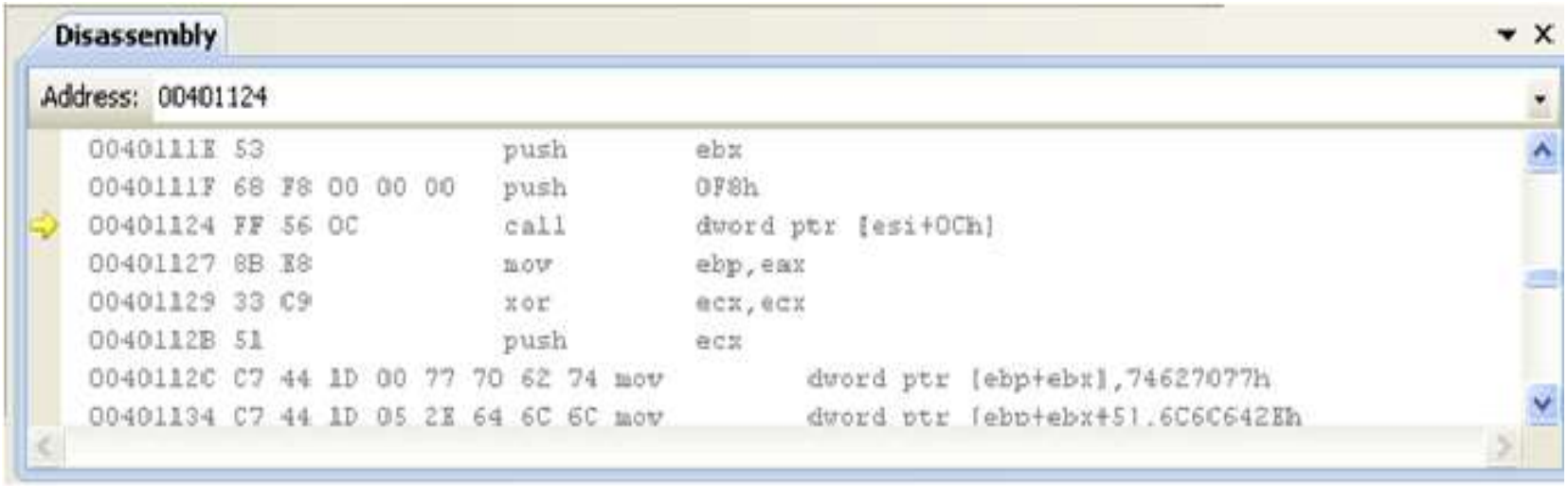


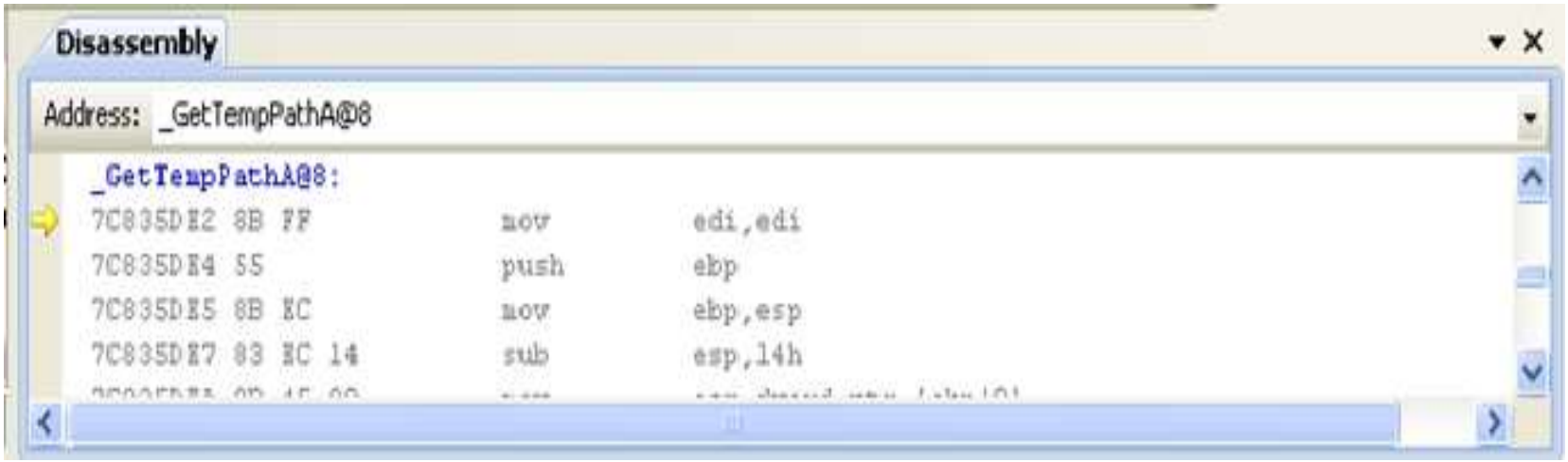
4. Execution

- Regsvr32:
  - The code block shown below will write “regsvr32 -s” to Memory. The “-s” option specifies regsvr32 to run silently and to not display any message boxes.

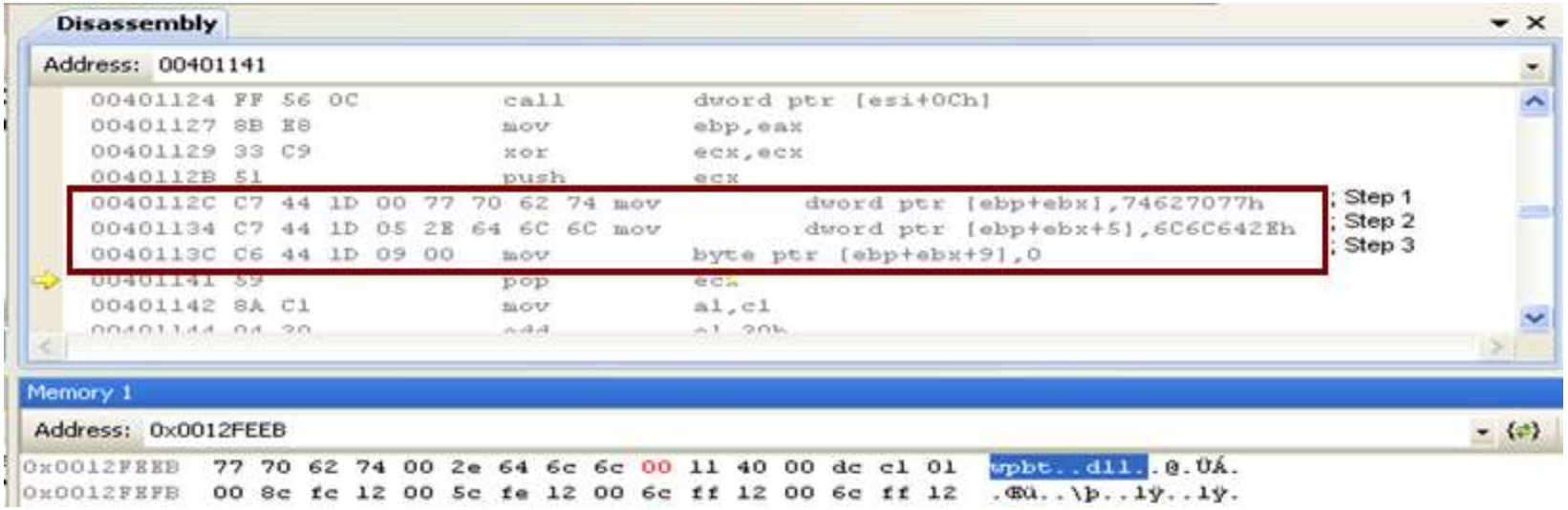


- Calling “GetTempPathA”:
- (ESI+0CH) is the starting address of the function “GetTempPathA”.

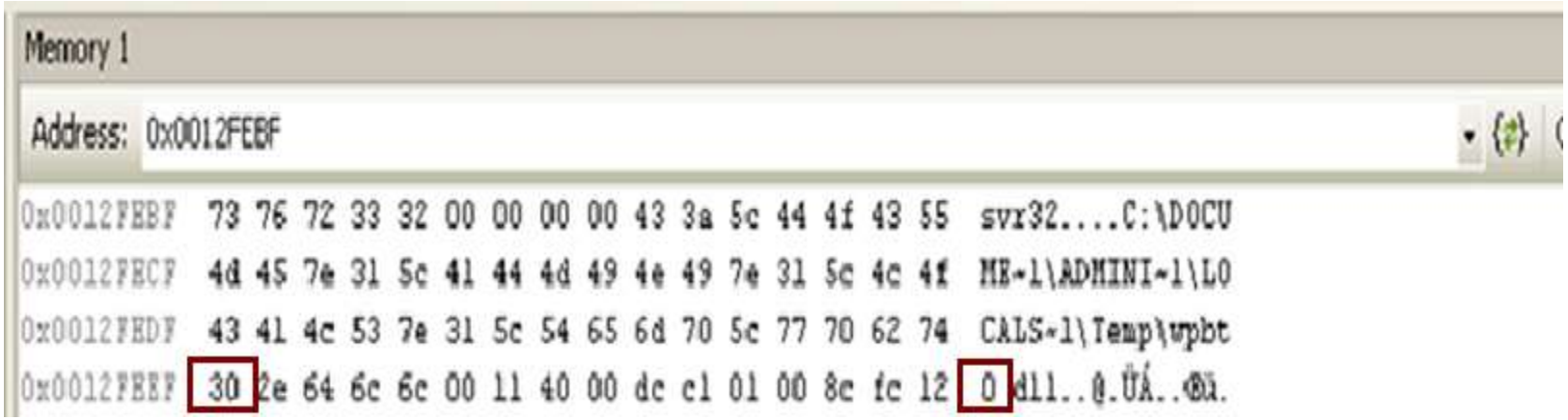




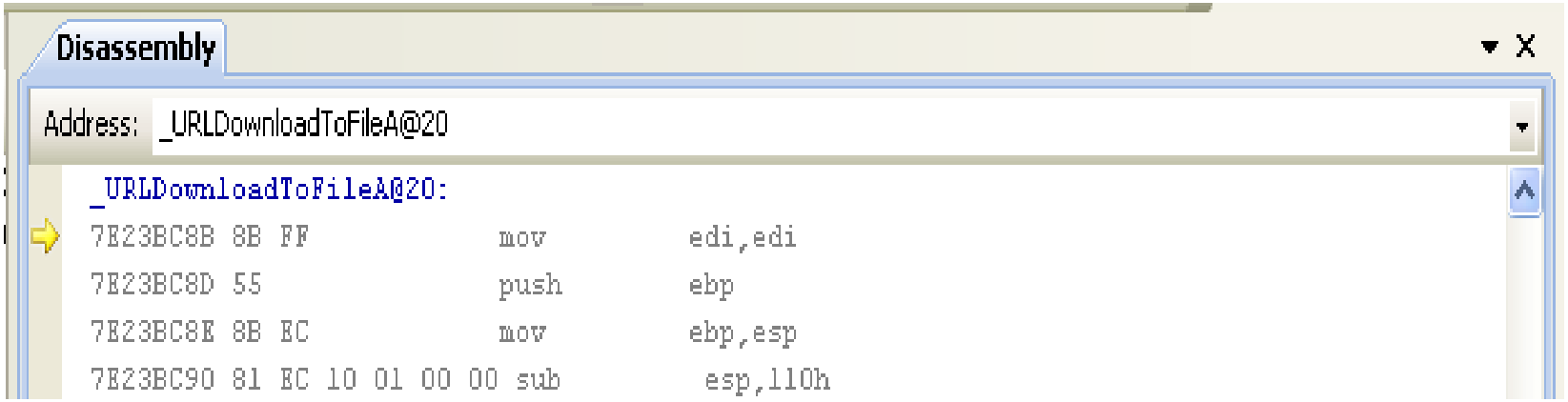
- Writing DLL:



- Renaming of DLL:



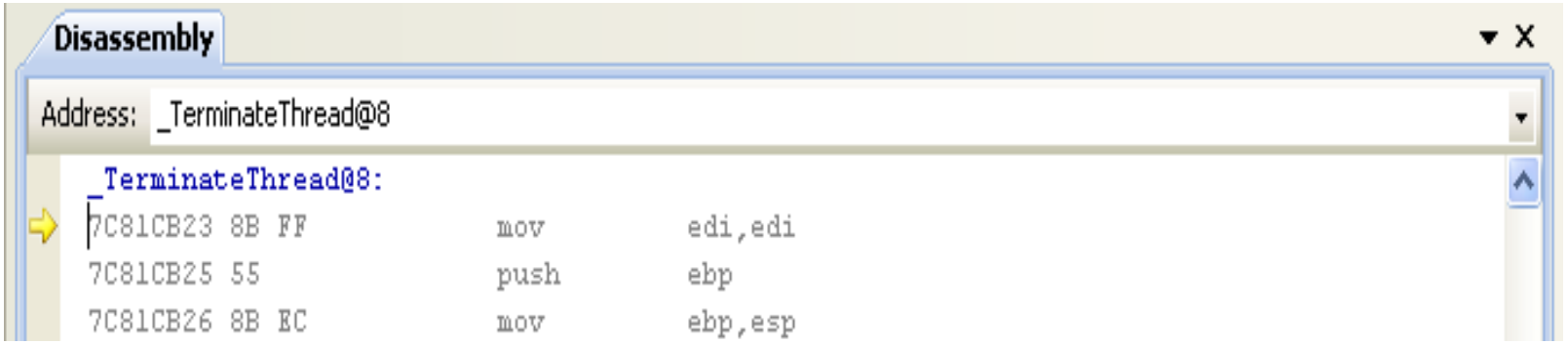
- Calling “UrlDownloadToFileA”:



- Subsequently, WinExec will be called on the downloaded file.

- Calling “TerminateThread”:

- The function will finally exit the native code.



Questions???

Feel free to drop mail @ "*Abhineet.Ayan.Verma@gmail.com*"


Thank You!