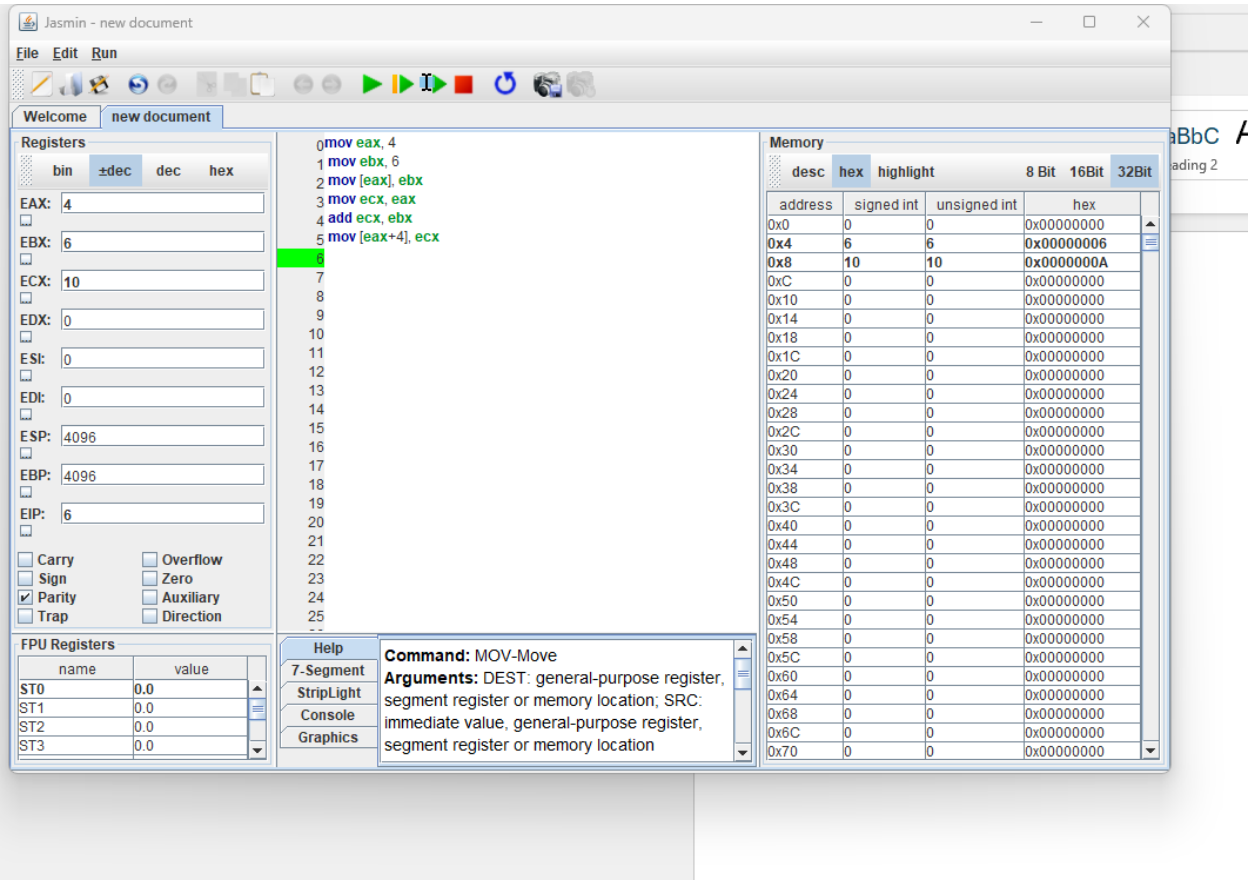
NGUYEN NAM KHANH – HE191159 – IA1902 – IAM302

LAB 9: Using Jasmin to run x86 Assembly Code

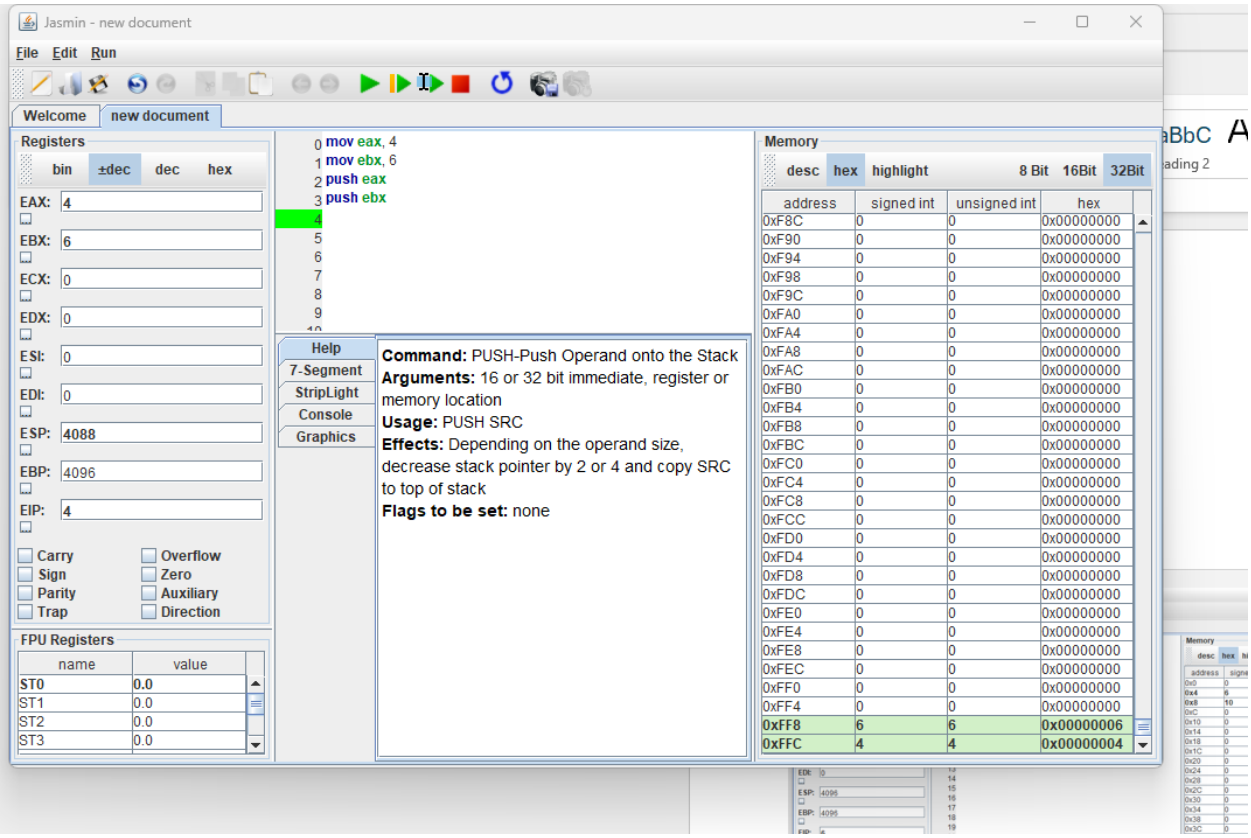
Storing Results in Memory

- EAX = 4
- EBX = 6
- ECX = 10
- Memory location 0x4 contains 6
- Memory location 0x8 contains 10



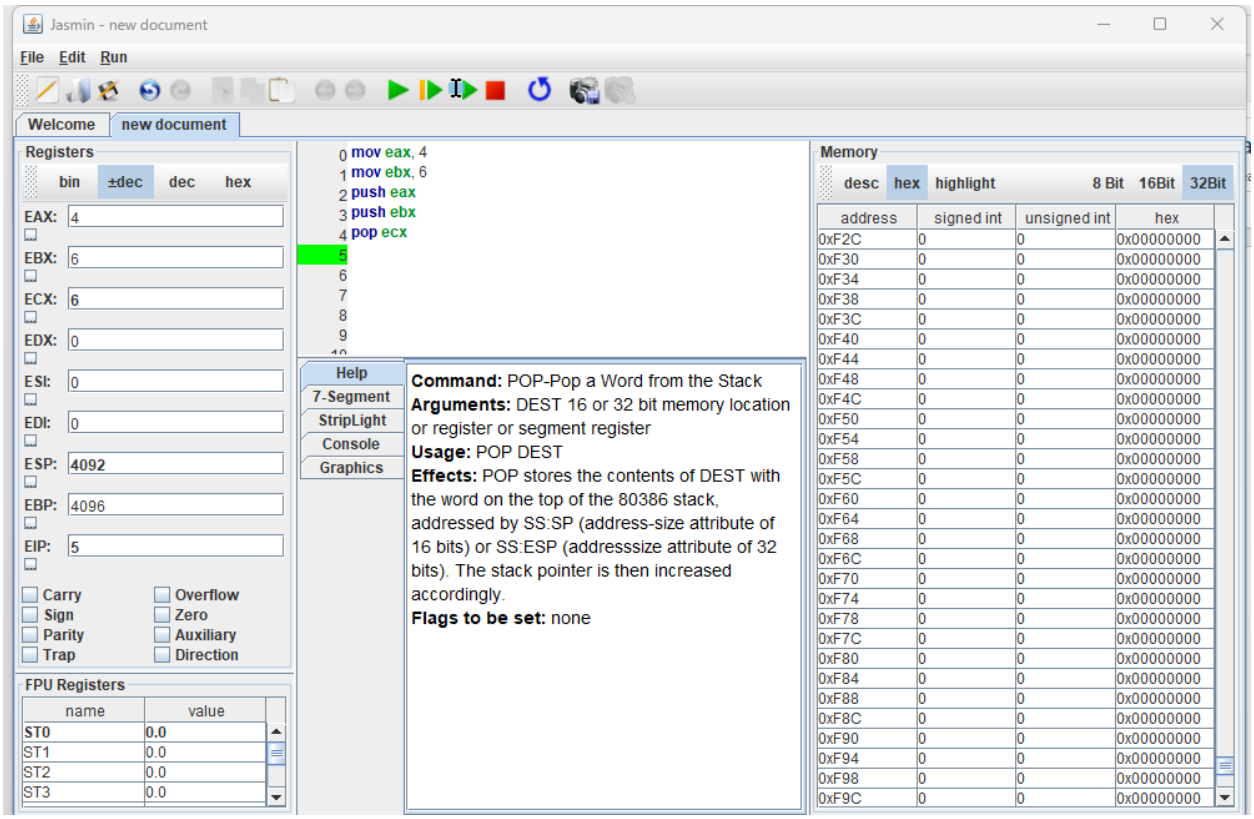
Understanding Push

- EAX contains 4
- EBX contains 6
- ESP contains 4088, which is 0xFF8, the new top of the stack.
- Memory location 0xFFC contains 4, the first value pushed onto the stack.
- Memory location 0xFF8 contains 6, the second value pushed onto the stack.



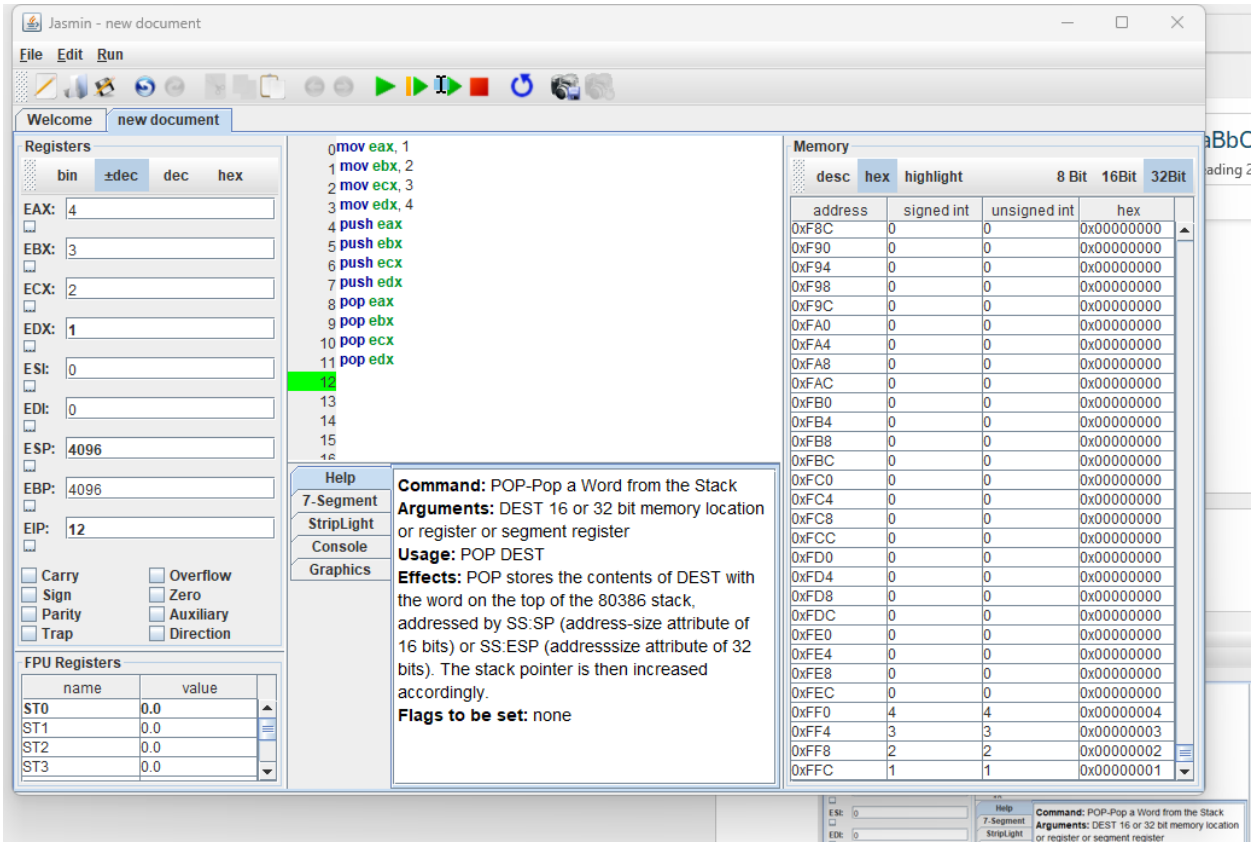
Understanding Pop

- ECX contains 6, the value popped off the top of the stack.
- ESP contains 4092, which is 0xFFC, the new top of the stack.
- Memory location 0xFFC contains 4, the first value pushed onto the stack.
- Memory location 0xFF8 contains 6, which is now the top value on the stack.

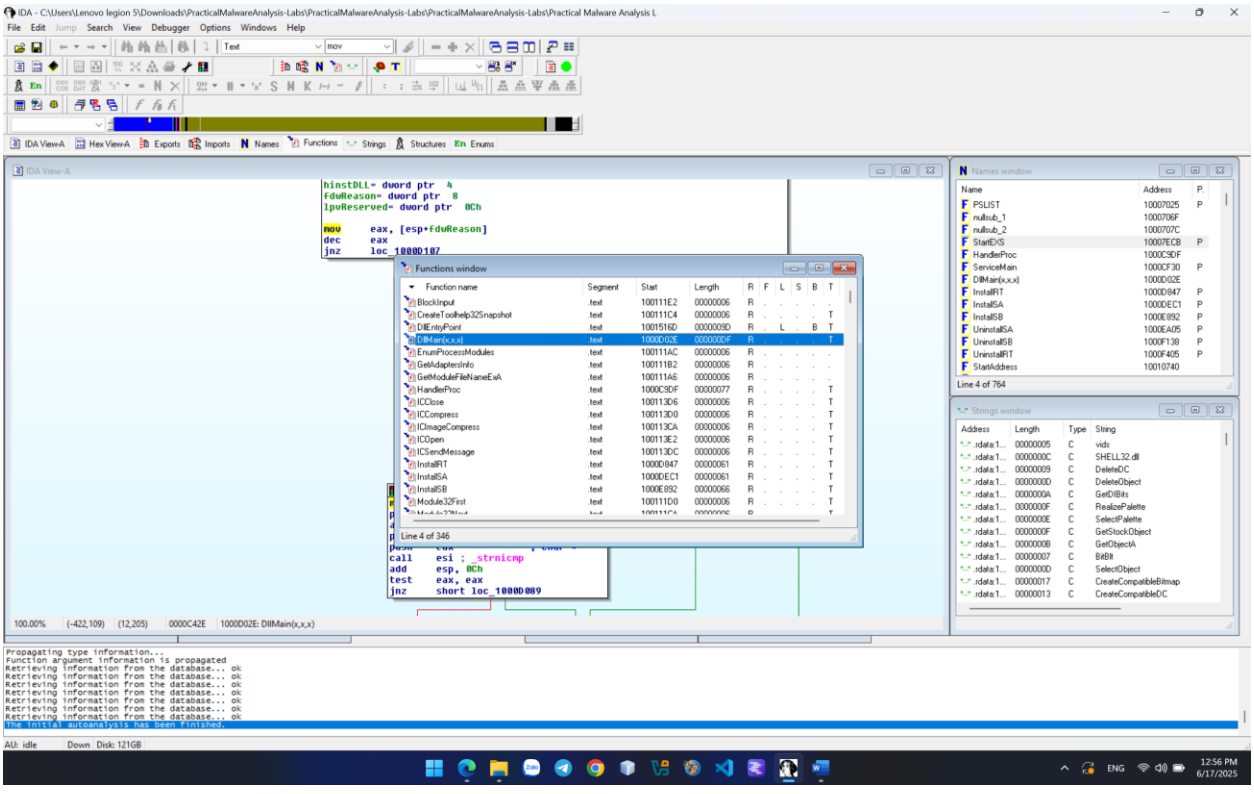


Reversing a Sequence

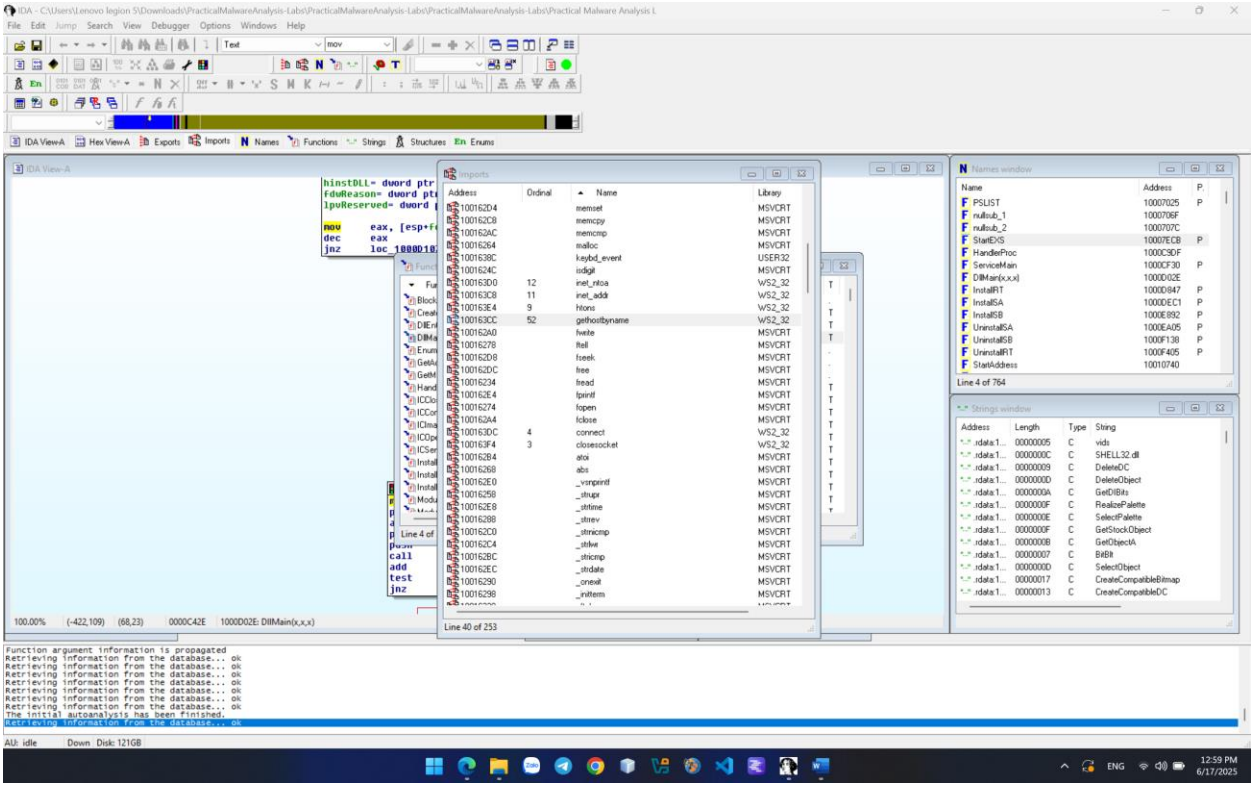
EAX = 4 EBX = 3 ECX = 2 EDX = 1



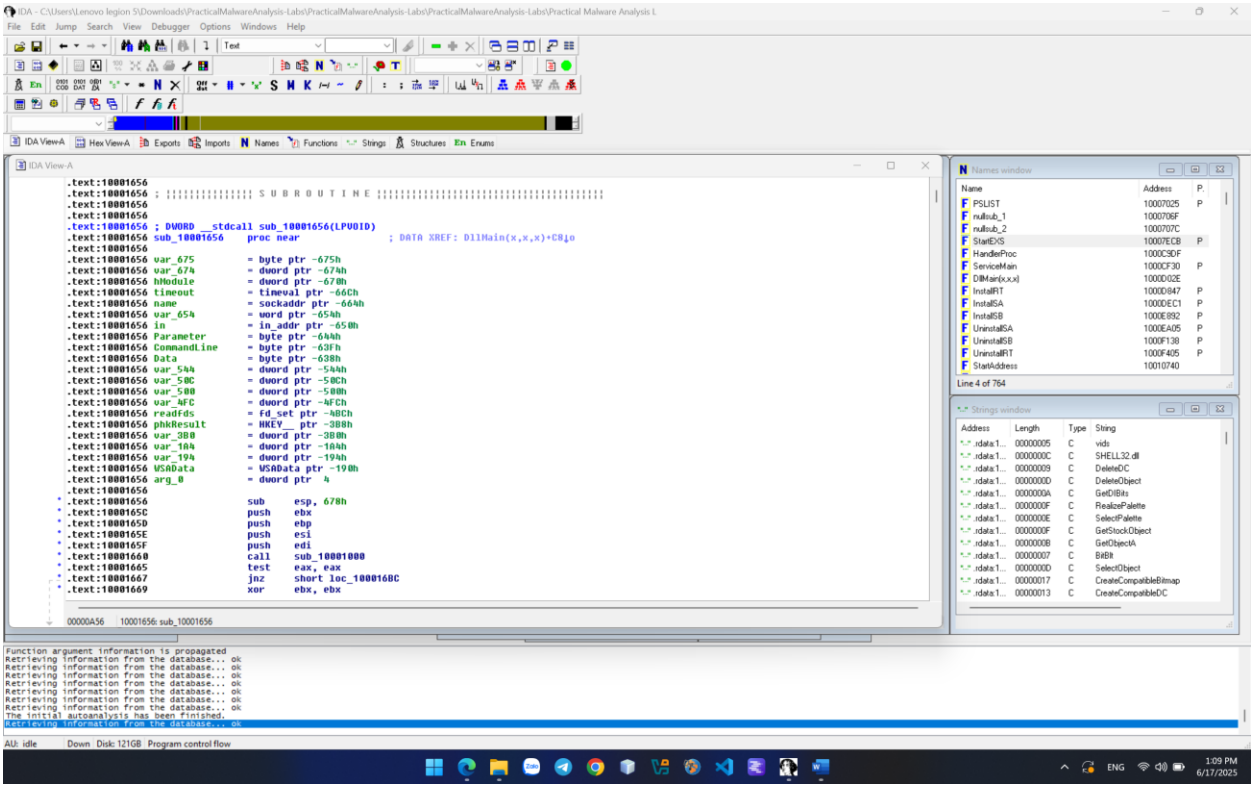
LAB 9.1: Lab05-01.dll in IDA Pro



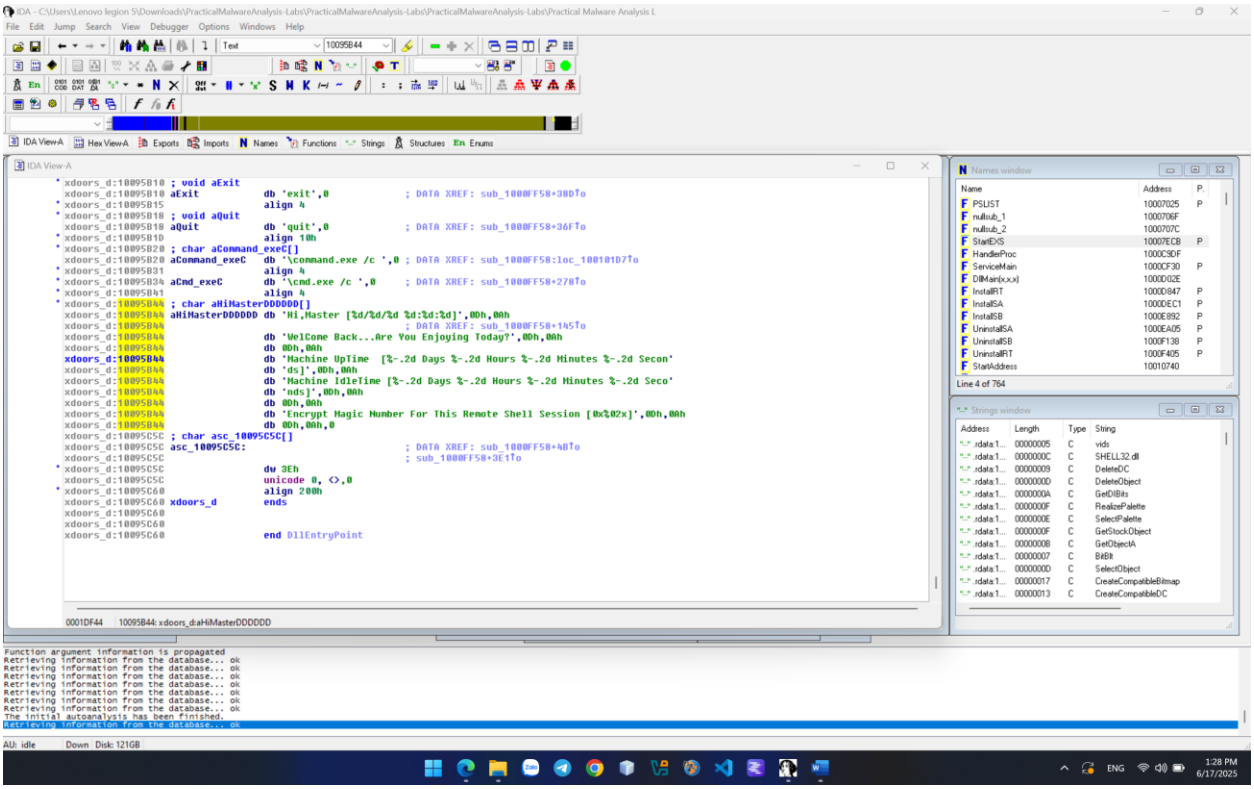
Find the import for gethostbyname



Count Local Variables for the Subroutine at 0x10001656

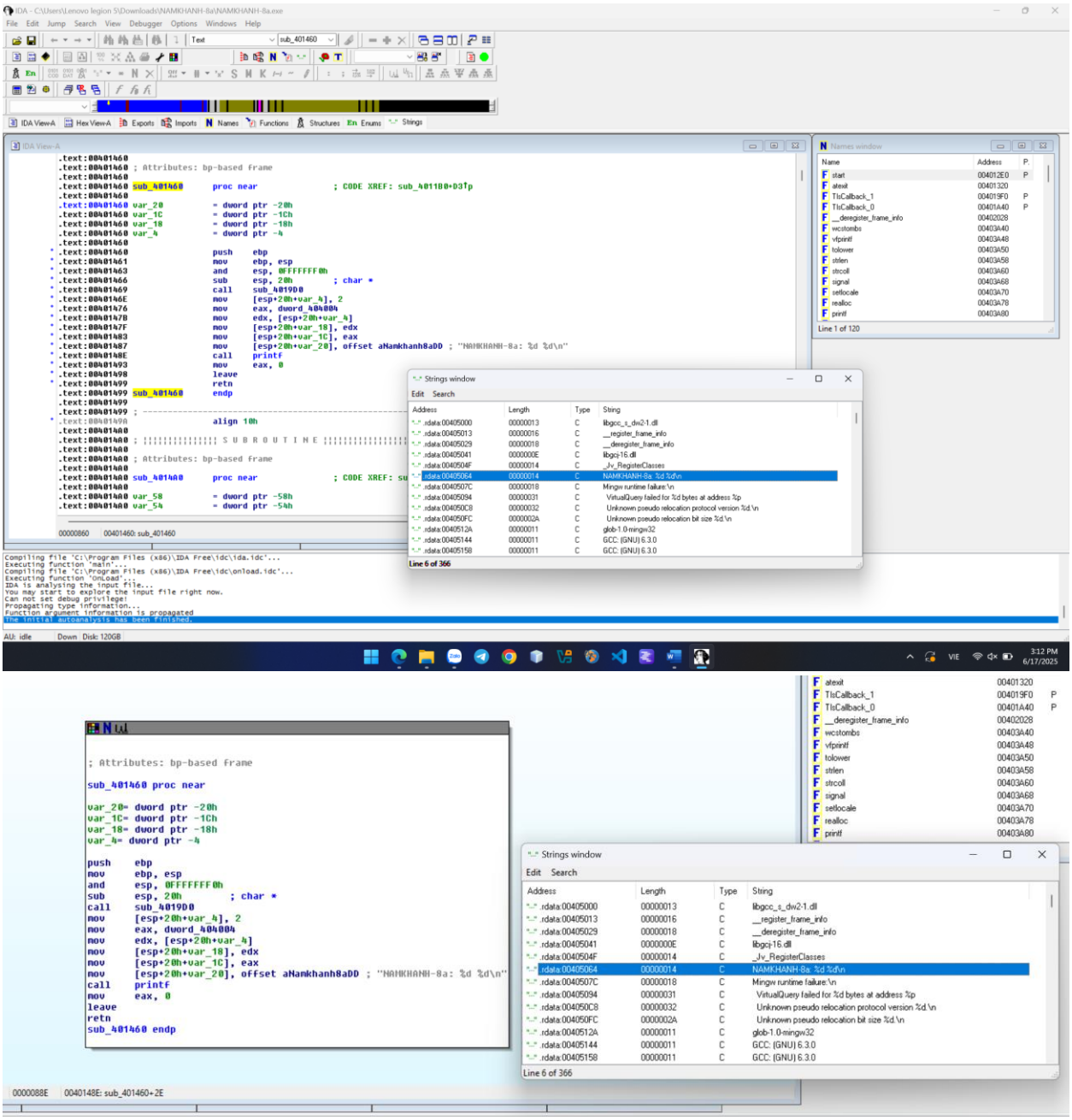


Finding the Purpose of the Code that References \cmd.exe /c



LAB 9.3: Disassembling C on Windows

Disassembling the EXE

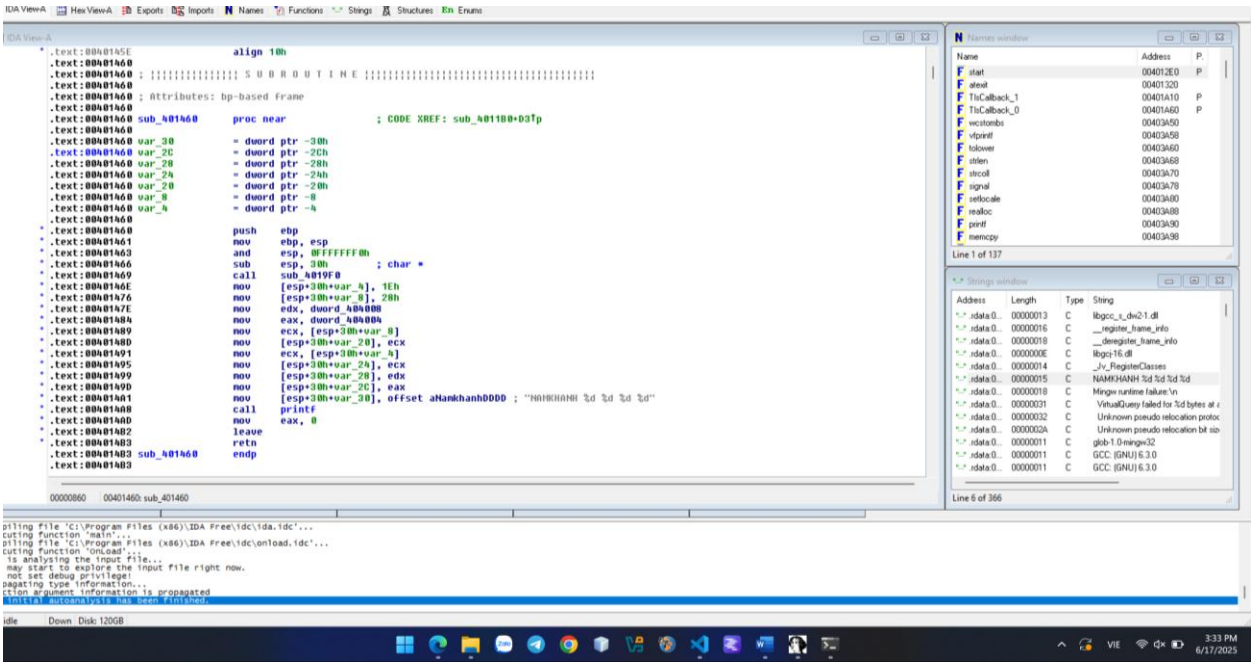


CHALLENGE:

Code:

```c
#include <stdio.h>

int x = 10;    // Biến toàn cục x
int m = 20;    // Biến toàn cục m

int main() {
    int y = 30;    // Biến cục bộ y
    int n = 40;    // Biến cục bộ thứ hai

    printf("NAMKHANH %d %d %d %d", x, m, y, n);  // Thay "NAMKHANH" bằng tên của bạn
    return 0;
}
```

Disassembling the EXE



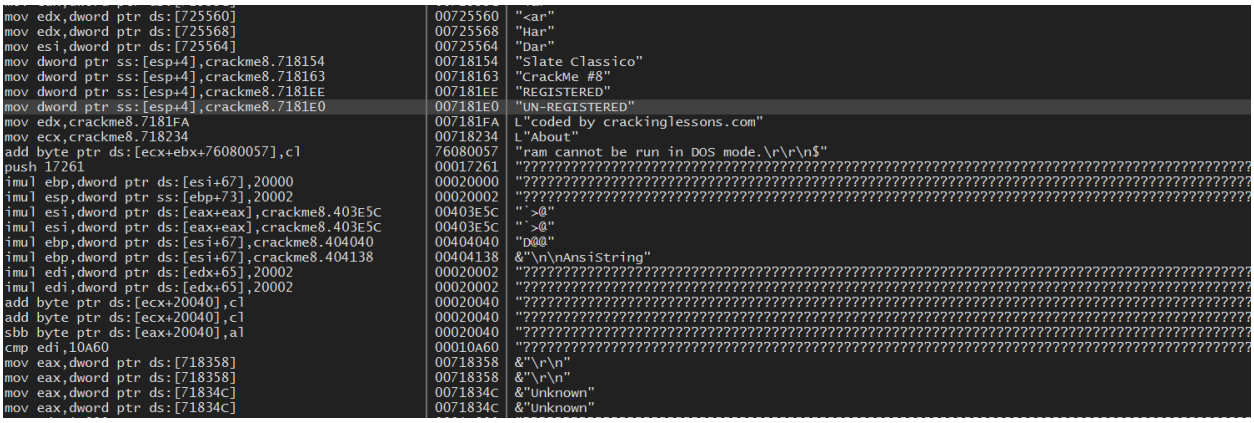

```
; Attributes: bp-based frame

sub_401460 proc near

var_30= dword ptr -30h
var_2C= dword ptr -2Ch
var_28= dword ptr -28h
var_24= dword ptr -24h
var_20= dword ptr -20h
var_8= dword ptr -8
var_4= dword ptr -4

push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF0h
sub     esp, 30h        ; char *
call    sub_4019F0
mov     [esp+30h+var_4], 1Eh
mov     [esp+30h+var_8], 28h
mov     edx, dword_404008
mov     eax, dword_404004
mov     ecx, [esp+30h+var_8]
mov     [esp+30h+var_20], ecx
mov     ecx, [esp+30h+var_4]
mov     [esp+30h+var_24], ecx
mov     [esp+30h+var_28], edx
mov     [esp+30h+var_2C], eax
mov     [esp+30h+var_30], offset aNamkhanhDDDD ; "NAMKHANH %d %d %d %d"
call    printf
mov     eax, 0
leave
retn
sub_401460 endp
```

CRACKME 8:

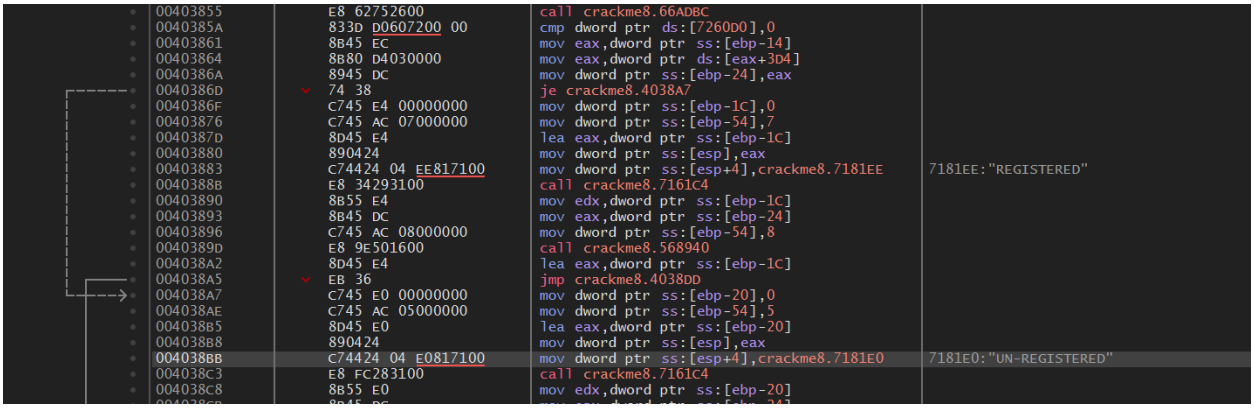
This crackme is for learning how to put hardware breakpoints on memory addresses and then patch it to register the program.

Extract the .exe file, run it for testing, and note down significant keywords. In this situation: UN-REGISTERED.

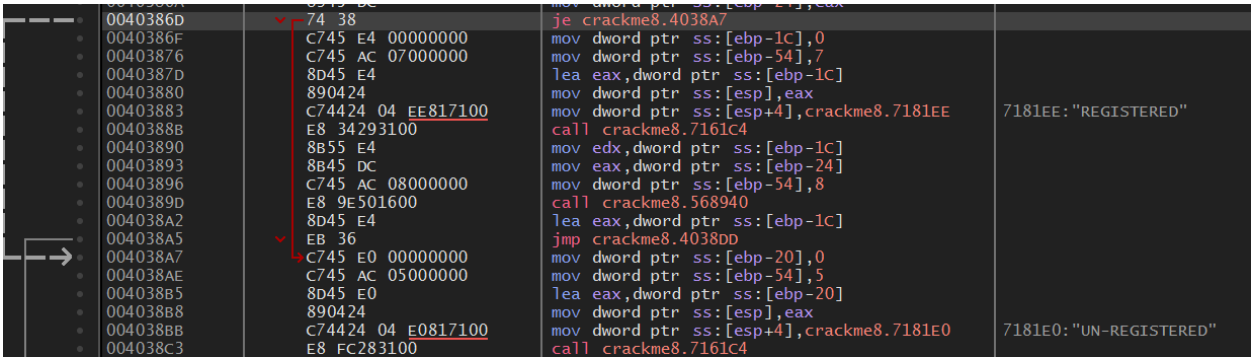
Then load the file into x32dbg, open the String references section to locate that keyword. Click on the keyword to find the corresponding code line.



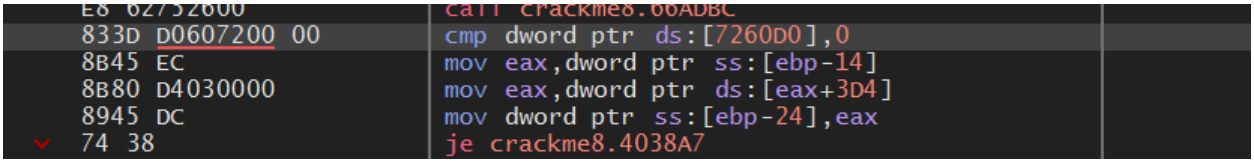
Pay attention to the code section above "UN-REGISTERED".



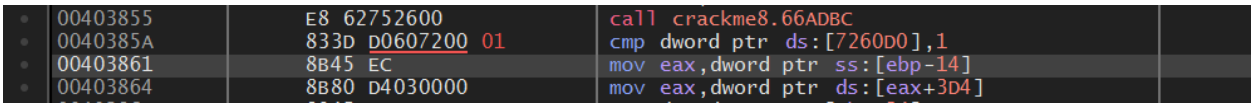
Locate the processing section that causes the program to display the "UN-REGISTERED" status. This is the code section shown below:



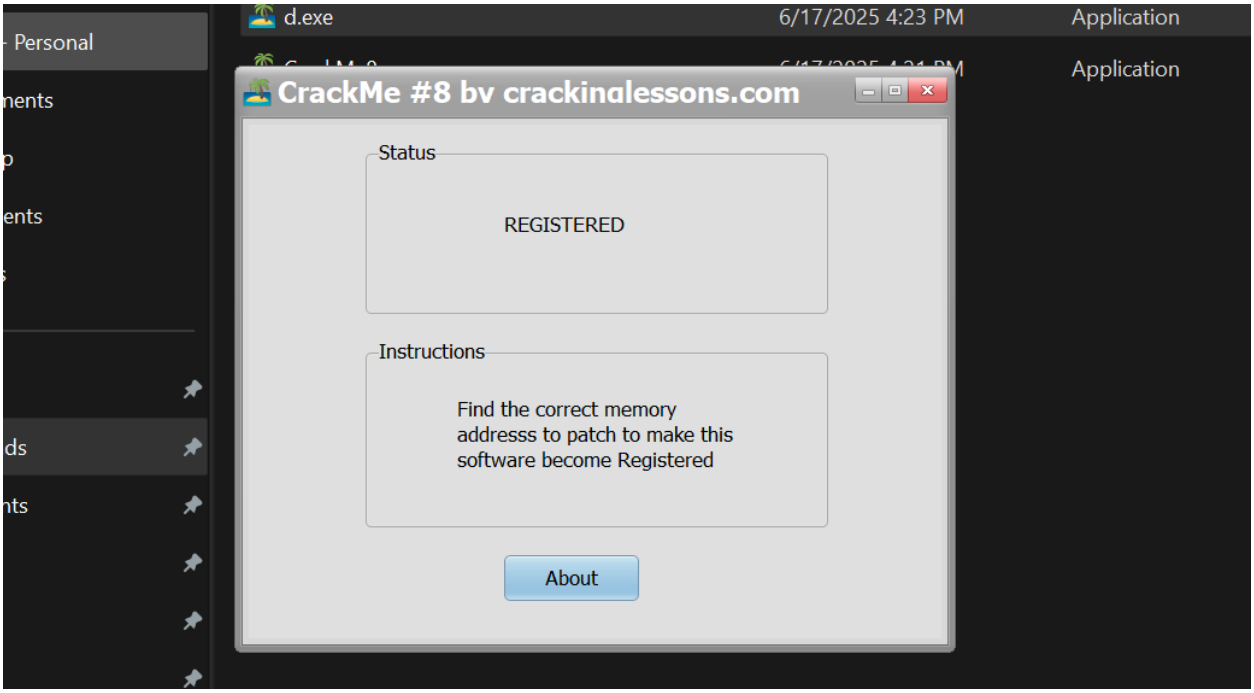
The JE instruction will jump to the "UN-REGISTERED" status processing section if the condition of the preceding CMP instruction is satisfied. Below are the details of the CMP instruction:



Compare the PTR DS variable with the value 0. If the condition is met, the JZ flag will be set and the JE instruction will be executed. To make the program display "REGISTERED" status instead of "UN-REGISTERED", we need to change the comparison value with the PTR DS variable to a non-zero value. Let's try changing it to 1.



Patching file and running again:



DONE!