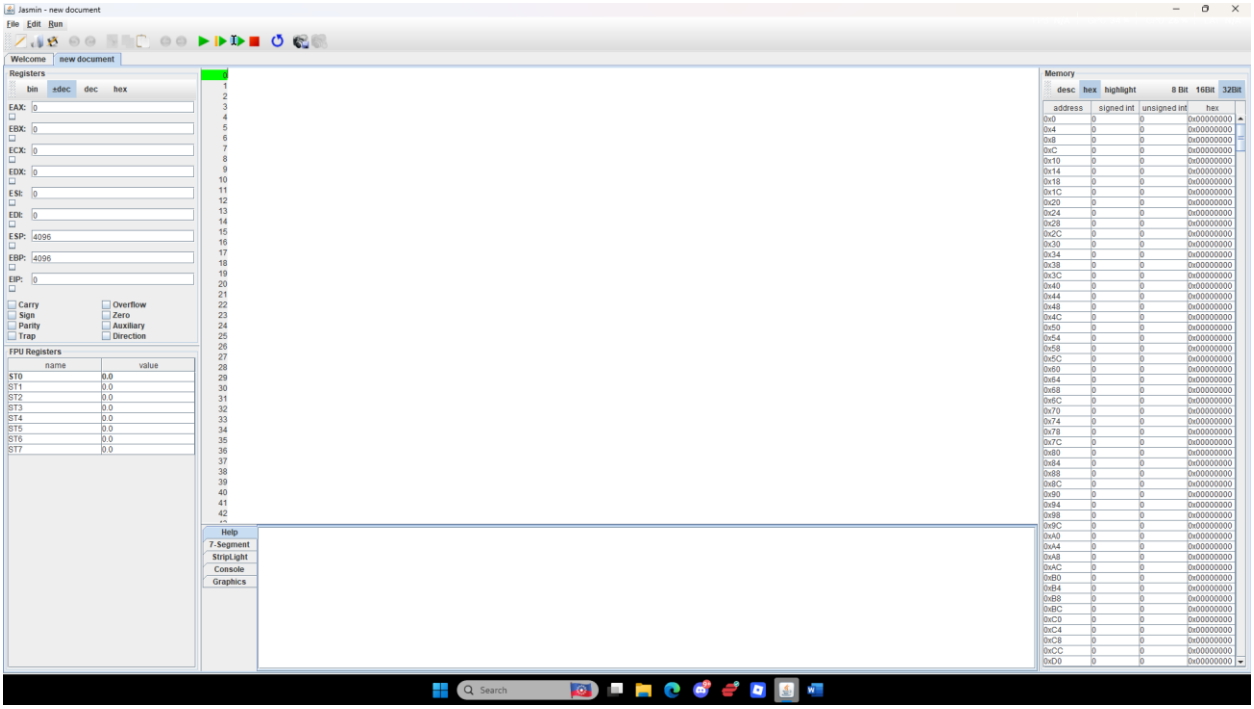
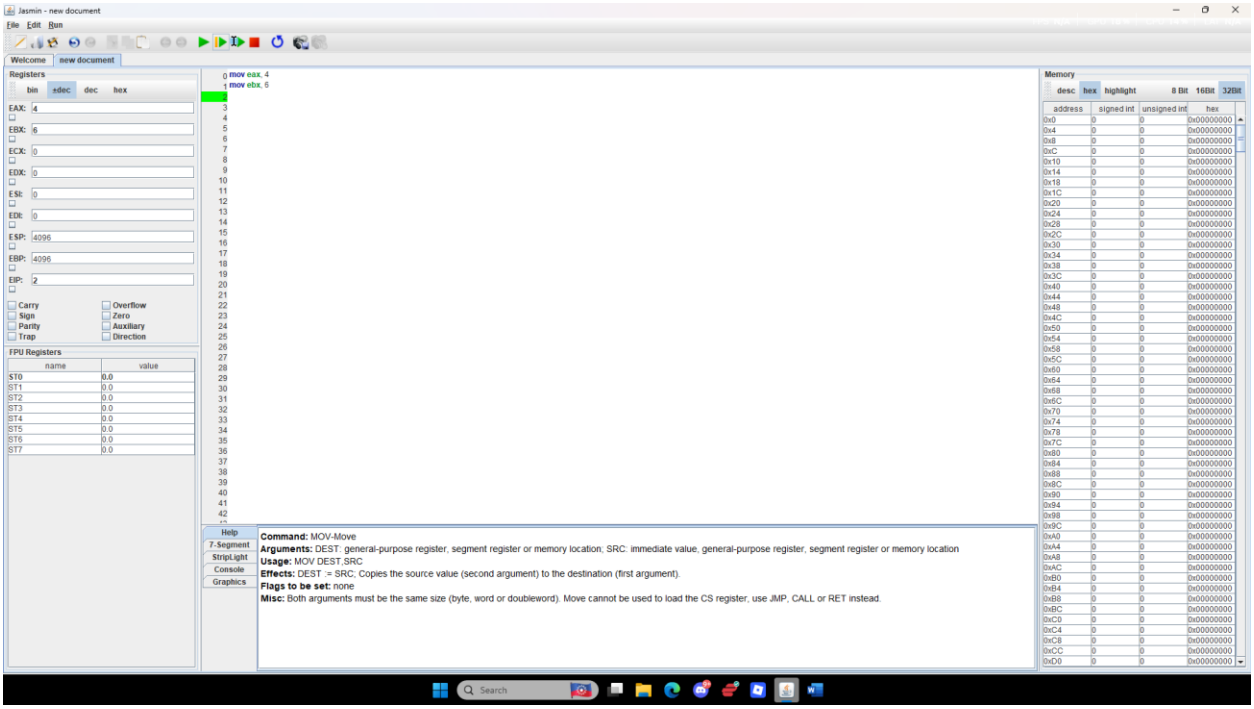
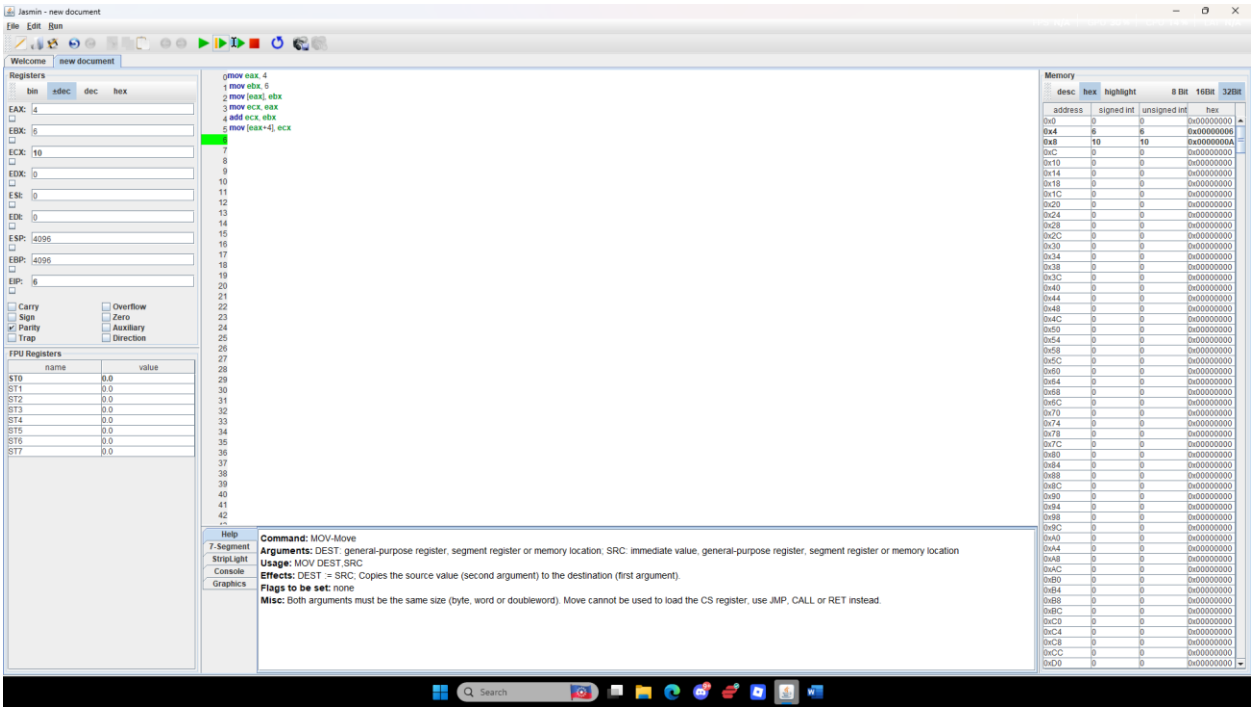
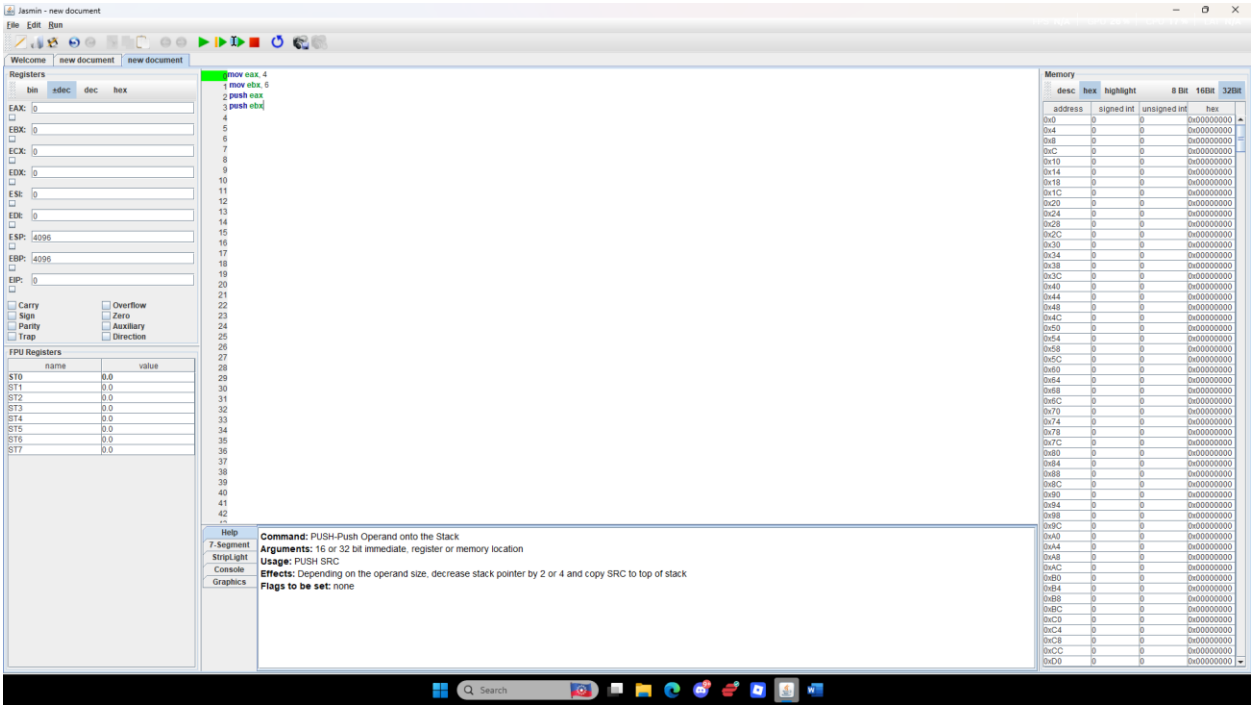
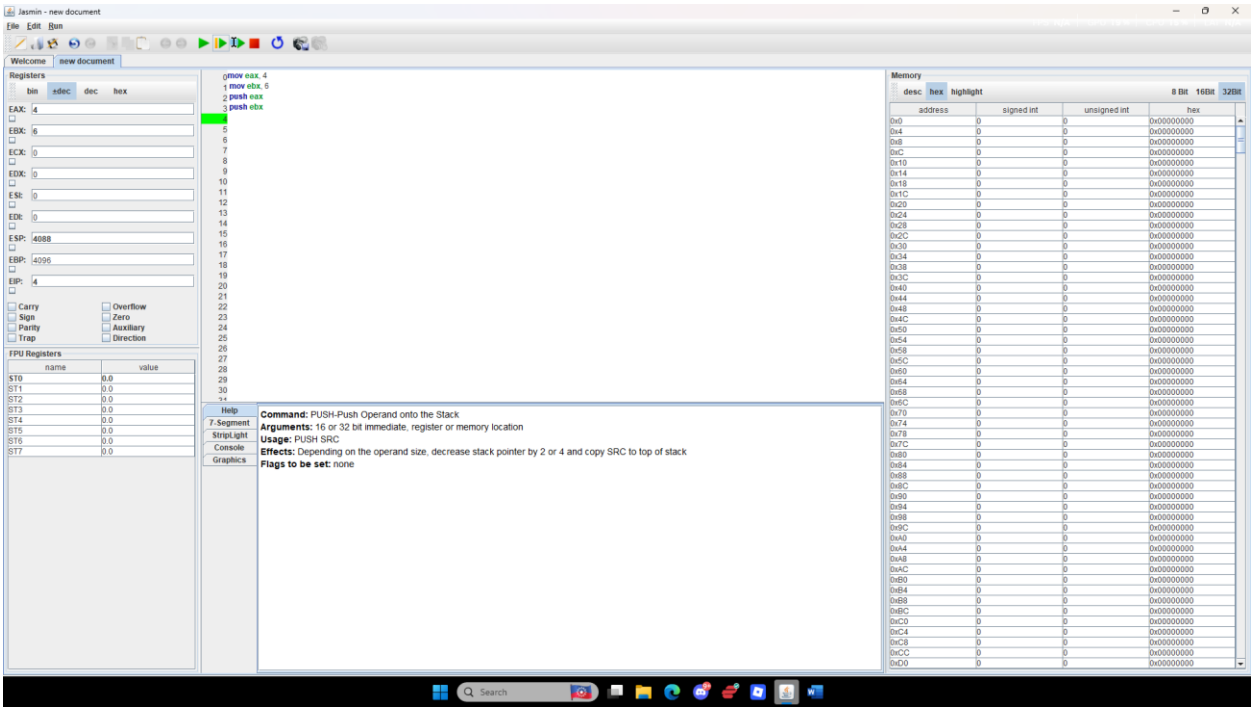
LAB 9

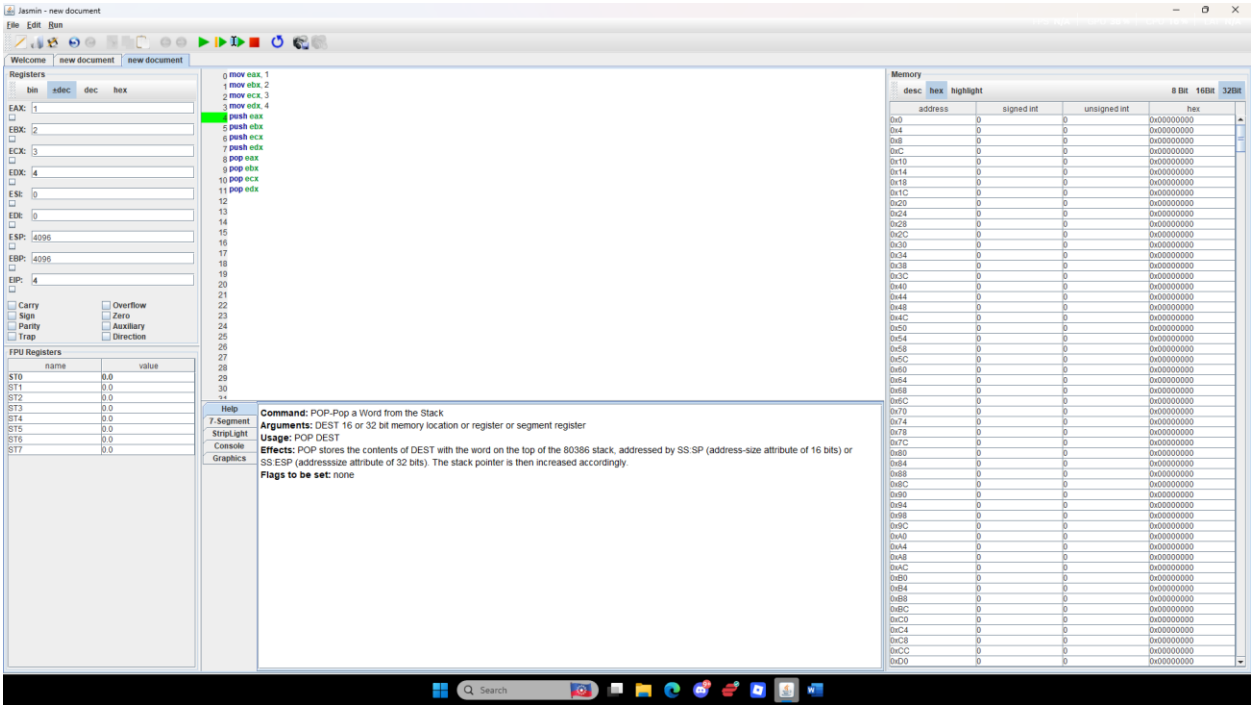

Jasmin - new document
File Edit Run
Welcome
new document
Registers
bin ±dec dec hex
EAX: 0
EBX: 0
ECX: 0
EDX: 0
ESI: 0
EDI: 0
ESP: 4096
EBP: 4096
EIP: 0
Carry
Sign
Parity
Trap
Overflow
Zero
Auxiliary
Direction
FPU Registers
name
value
ST0 0.0
ST1 0.0
ST2 0.0
ST3 0.0
ST4 0.0
ST5 0.0
ST6 0.0
ST7 0.0
Help
7-Segment
StripLight
Console
Graphics
Memory
desc hex highlight 8 Bit 16Bit 32Bit
address
signed int
unsigned int
hex

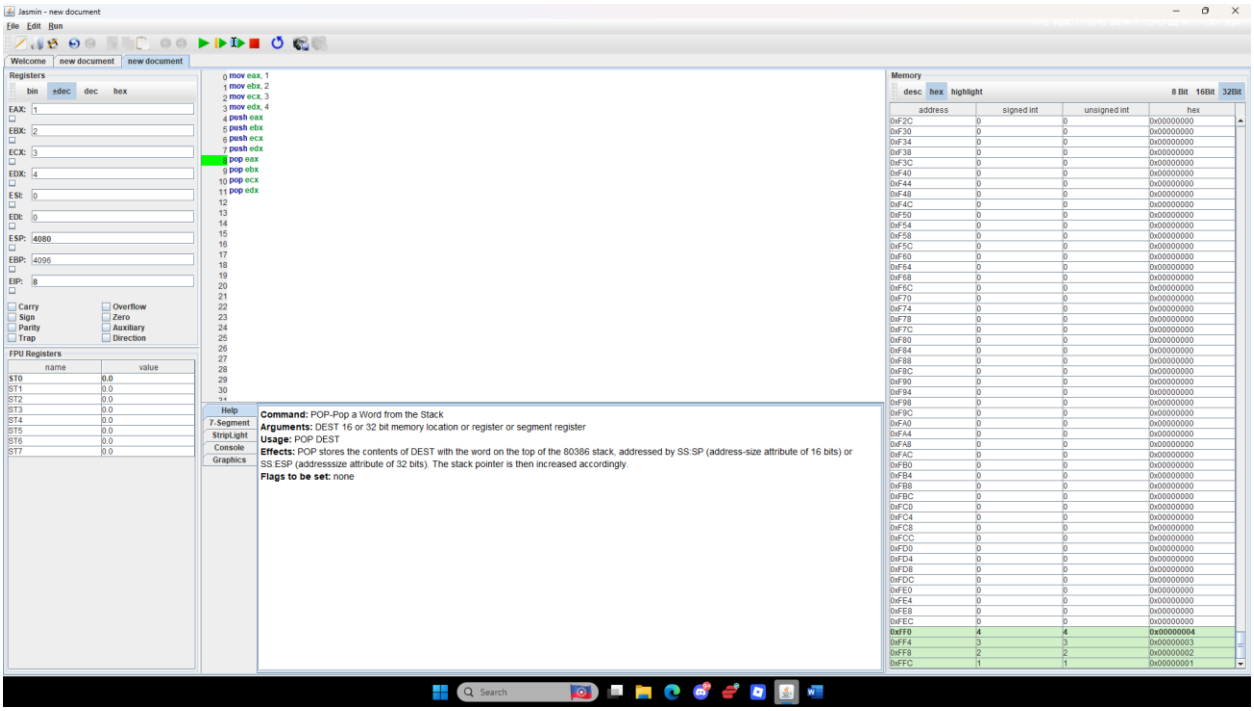

Jasmin - new document
File Edit Run
Welcome
new document
Registers
bin ±dec dec hex
EAX: 4
EBX: 6
ECX: 0
EDX: 0
ESI: 0
EDI: 0
ESP: 4096
EBP: 4096
EIP: 2
Carry
Sign
Parity
Trap
Overflow
Zero
Auxiliary
Direction
FPU Registers
name
value
ST0 0.0
ST1 0.0
ST2 0.0
ST3 0.0
ST4 0.0
ST5 0.0
ST6 0.0
ST7 0.0
0 mov eax, 4
1 mov ebx, 6
Help
7-Segment
StripLight
Console
Graphics
Command: MOV-Move
Arguments: DEST: general-purpose register, segment register or memory location; SRC: immediate value, general-purpose register, segment register or memory location
Usage: MOV DEST,SRC
Effects: DEST := SRC; Copies the source value (second argument) to the destination (first argument).
Flags to be set: none
Misc: Both arguments must be the same size (byte, word or doubleword). Move cannot be used to load the CS register, use JMP, CALL or RET instead.
Memory
desc hex highlight 8 Bit 16Bit 32Bit
address
signed int
unsigned int
hex

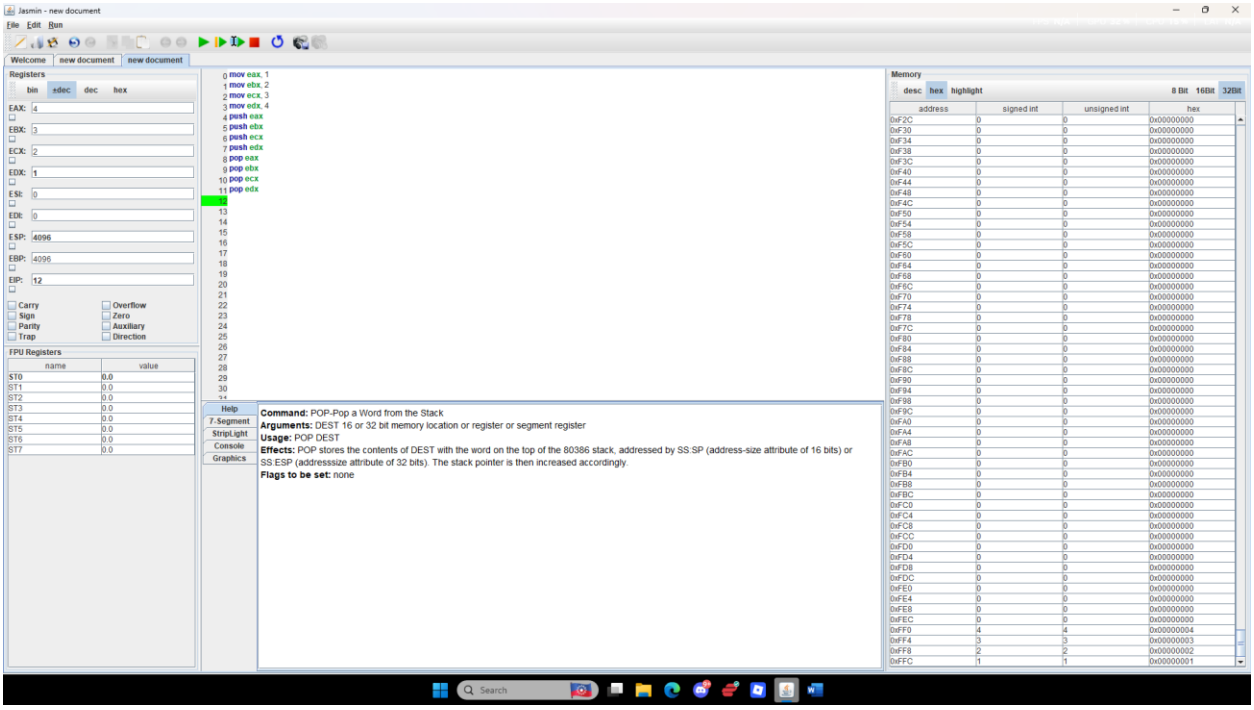

Jasmin - new document
File Edit Run
Welcome
new document
Registers
bin ±dec dec hex
EAX: 4
EBX: 6
ECX: 10
EDX: 0
ESI: 0
EDI: 0
ESP: 4096
EBP: 4096
EIP: 6
Carry
Sign
Parity
Trap
Overflow
Zero
Auxiliary
Direction
FPU Registers
name
value
ST0 0.0
ST1 0.0
ST2 0.0
ST3 0.0
ST4 0.0
ST5 0.0
ST6 0.0
ST7 0.0
0 mov eax, 4
1 mov ebx, 6
2 mov [eax], ebx
3 mov ecx, eax
4 add ecx, ebx
5 mov [eax+4], ecx
Help
7-Segment
StripLight
Console
Graphics
Command: MOV-Move
Arguments: DEST: general-purpose register, segment register or memory location; SRC: immediate value, general-purpose register, segment register or memory location
Usage: MOV DEST,SRC
Effects: DEST := SRC; Copies the source value (second argument) to the destination (first argument).
Flags to be set: none
Misc: Both arguments must be the same size (byte, word or doubleword). Move cannot be used to load the CS register; use JMP, CALL or RET instead.
Memory
desc hex highlight 8 Bit 16Bit 32Bit
address signed int unsigned int hex
0x0 0 0 0x00000000
0x4 6 6 0x00000006
0x8 10 10 0x0000000A
Search

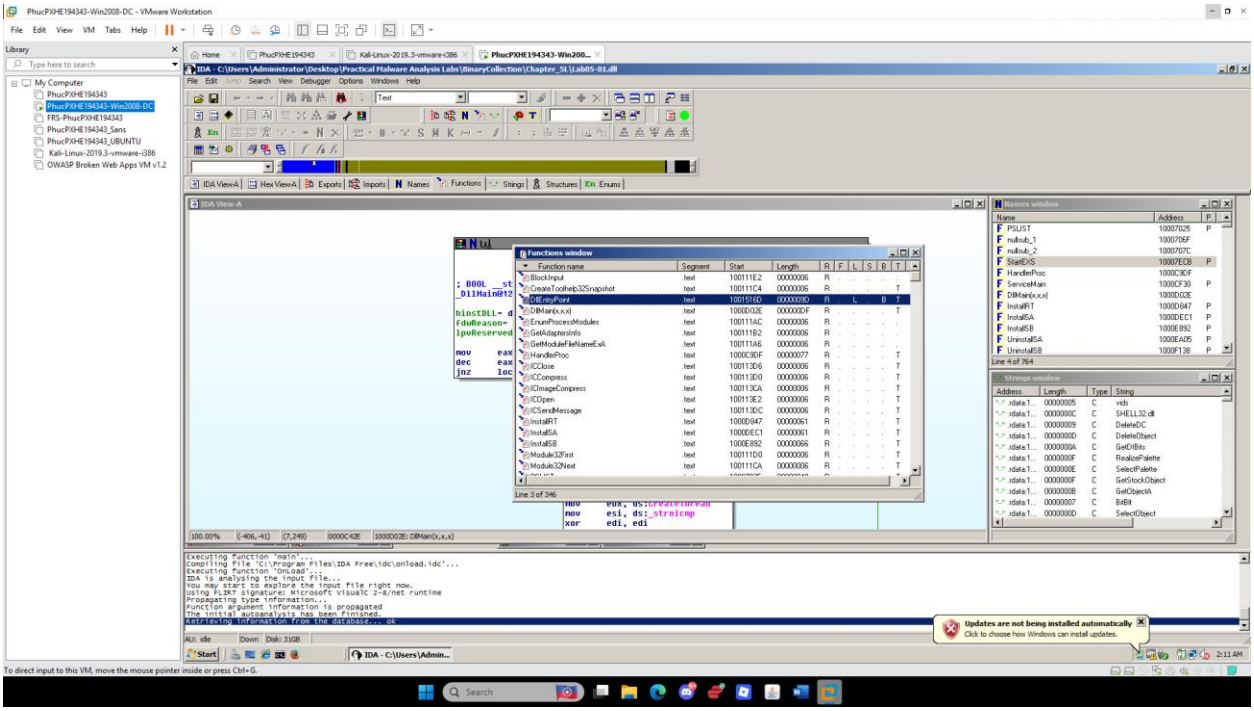

Jasmin - new document
File Edit Run
Welcome
new document
new document
Registers
bin ±dec dec hex
EAX: 0
EBX: 0
ECX: 0
EDX: 0
ESI: 0
EDI: 0
ESP: 4096
EBP: 4096
EIP: 0
Carry
Sign
Parity
Trap
Overflow
Zero
Auxiliary
Direction
FPU Registers
name
value
ST0 0.0
ST1 0.0
ST2 0.0
ST3 0.0
ST4 0.0
ST5 0.0
ST6 0.0
ST7 0.0
0 mov eax, 4
1 mov ebx, 6
2 push eax
3 push ebx
Help
7-Segment
StripLight
Console
Graphics
Command: PUSH-Push Operand onto the Stack
Arguments: 16 or 32 bit immediate, register or memory location
Usage: PUSH SRC
Effects: Depending on the operand size, decrease stack pointer by 2 or 4 and copy SRC to top of stack
Flags to be set: none
Memory
desc hex highlight 8 Bit 16Bit 32Bit
address signed int unsigned int hex
Search

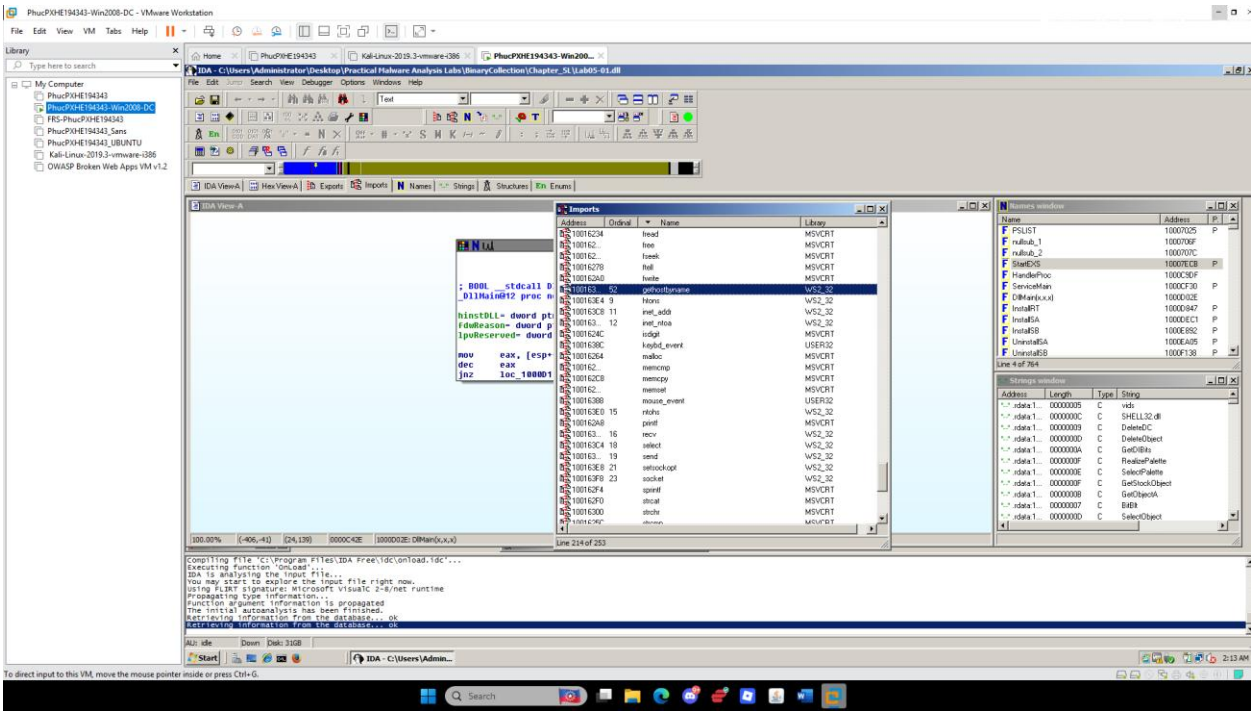

Jasmin - new document
File Edit Run
Welcome
new document
Registers
bin ±dec dec hex
EAX: 4
EBX: 6
ECX: 0
EDX: 0
ESI: 0
EDI: 0
ESP: 4088
EBP: 4096
EIP: 4
Carry
Sign
Parity
Trap
Overflow
Zero
Auxiliary
Direction
FPU Registers
name
value
ST0 0.0
ST1 0.0
ST2 0.0
ST3 0.0
ST4 0.0
ST5 0.0
ST6 0.0
ST7 0.0
0 mov eax, 4
1 mov ebx, 6
2 push eax
3 push ebx
Help
7-Segment
StripLight
Console
Graphics
Command: PUSH-Push Operand onto the Stack
Arguments: 16 or 32 bit immediate, register or memory location
Usage: PUSH SRC
Effects: Depending on the operand size, decrease stack pointer by 2 or 4 and copy SRC to top of stack
Flags to be set: none
Memory
desc hex highlight
8 Bit 16Bit 32Bit
address
signed int
unsigned int
hex
Search

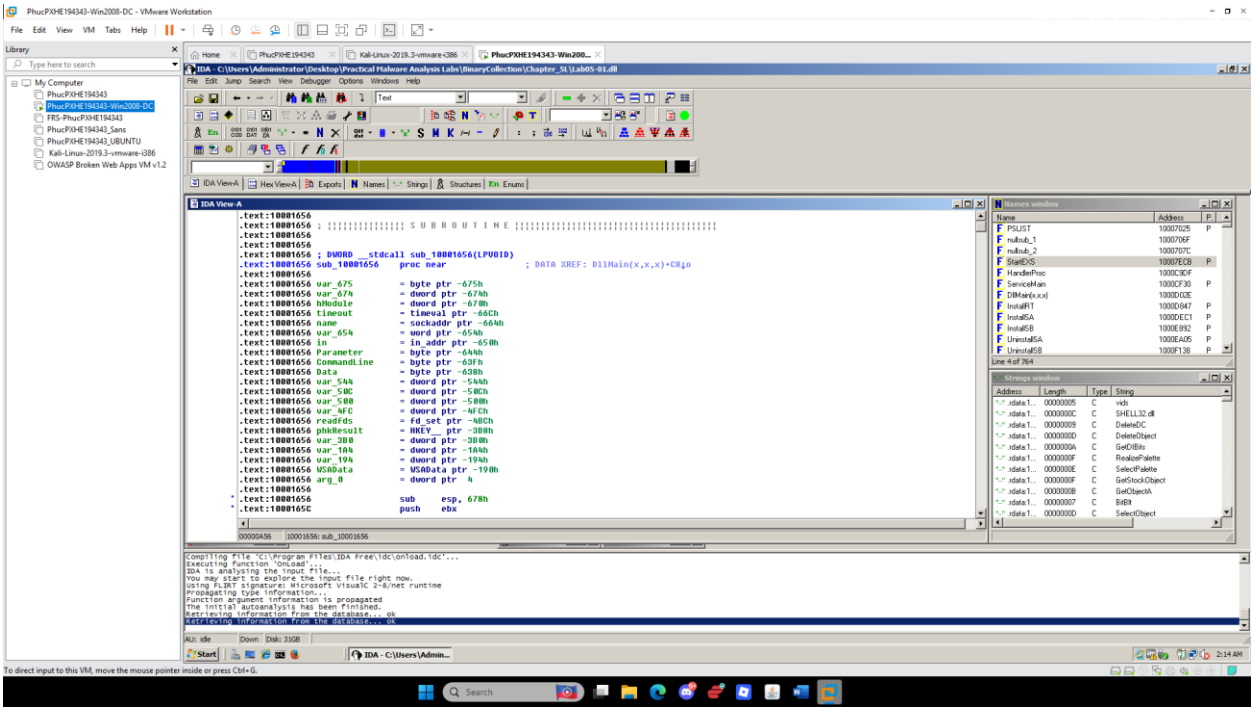

Jasmin - new document
File Edit Run
Welcome
new document
new document
Registers
bin ±dec dec hex
EAX: 1
EBX: 2
ECX: 3
EDX: 4
ESI: 0
EDI: 0
ESP: 4096
EBP: 4096
EIP: 4
Carry
Sign
Parity
Trap
Overflow
Zero
Auxiliary
Direction
FPU Registers
name
value
ST0 0.0
ST1 0.0
ST2 0.0
ST3 0.0
ST4 0.0
ST5 0.0
ST6 0.0
ST7 0.0
0 mov eax, 1
1 mov ebx, 2
2 mov ecx, 3
3 mov edx, 4
4 push eax
5 push ebx
6 push ecx
7 push edx
8 pop eax
9 pop ebx
10 pop ecx
11 pop edx
Help
7-Segment
StripLight
Console
Graphics
Command: POP-Pop a Word from the Stack
Arguments: DEST 16 or 32 bit memory location or register or segment register
Usage: POP DEST
Effects: POP stores the contents of DEST with the word on the top of the 80386 stack, addressed by SS:SP (address-size attribute of 16 bits) or SS:ESP (addresssize attribute of 32 bits). The stack pointer is then increased accordingly.
Flags to be set: none
Memory
desc hex highlight
8 Bit 16Bit 32Bit
address
signed int
unsigned int
hex
Search

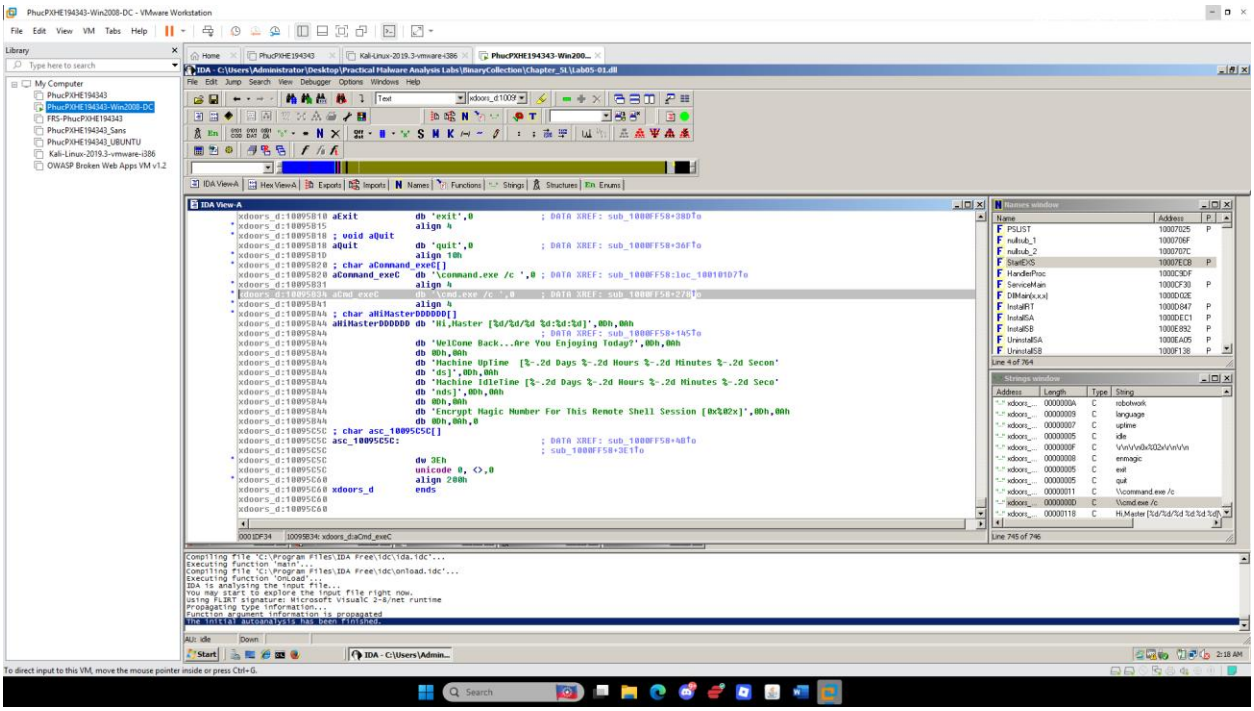

Jasmin - new document
File Edit Run
Welcome
new document
new document
Registers
bin ±dec dec hex
EAX: 1
EBX: 2
ECX: 3
EDX: 4
ESI: 0
EDI: 0
ESP: 4080
EBP: 4096
EIP: 8
Carry
Sign
Parity
Trap
Overflow
Zero
Auxiliary
Direction
FPU Registers
name
value
ST0 0.0
ST1 0.0
ST2 0.0
ST3 0.0
ST4 0.0
ST5 0.0
ST6 0.0
ST7 0.0
0 mov eax, 1
1 mov ebx, 2
2 mov ecx, 3
3 mov edx, 4
4 push eax
5 push ebx
6 push ecx
7 push edx
8 pop eax
9 pop ebx
10 pop ecx
11 pop edx
Help
7-Segment
StripLight
Console
Graphics
Command: POP-Pop a Word from the Stack
Arguments: DEST 16 or 32 bit memory location or register or segment register
Usage: POP DEST
Effects: POP stores the contents of DEST with the word on the top of the 80386 stack, addressed by SS:SP (address-size attribute of 16 bits) or SS:ESP (addresssize attribute of 32 bits). The stack pointer is then increased accordingly.
Flags to be set: none
Memory
desc hex highlight
8 Bit 16Bit 32Bit
address
signed int
unsigned int
hex
0xFF0 4 4 0x00000004
0xFF4 3 3 0x00000003
0xFF8 2 2 0x00000002
0xFFC 1 1 0x00000001
Search

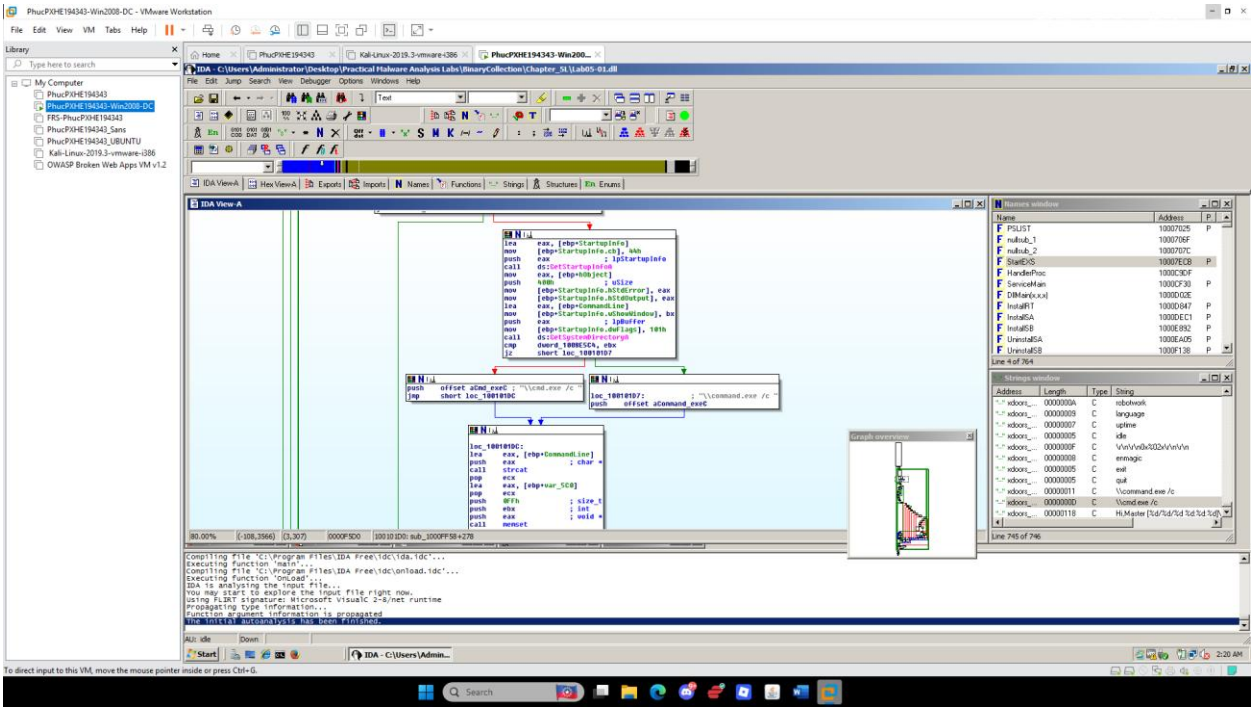

Jasmin - new document
File Edit Run
Welcome
new document
new document
Registers
bin ±dec dec hex
EAX: 4
EBX: 3
ECX: 2
EDX: 1
ESI: 0
EDI: 0
ESP: 4096
EBP: 4096
EIP: 12
Carry
Sign
Parity
Trap
Overflow
Zero
Auxiliary
Direction
FPU Registers
name
value
ST0 0.0
ST1 0.0
ST2 0.0
ST3 0.0
ST4 0.0
ST5 0.0
ST6 0.0
ST7 0.0
0 mov eax, 1
1 mov ebx, 2
2 mov ecx, 3
3 mov edx, 4
4 push eax
5 push ebx
6 push ecx
7 push edx
8 pop eax
9 pop ebx
10 pop ecx
11 pop edx
Help
7-Segment
StripLight
Console
Graphics
Command: POP-Pop a Word from the Stack
Arguments: DEST 16 or 32 bit memory location or register or segment register
Usage: POP DEST
Effects: POP stores the contents of DEST with the word on the top of the 80386 stack, addressed by SS:SP (address-size attribute of 16 bits) or SS:ESP (addresssize attribute of 32 bits). The stack pointer is then increased accordingly.
Flags to be set: none
Memory
desc hex highlight
8 Bit 16Bit 32Bit
address
signed int
unsigned int
hex
0xFF0 4 4 0x00000004
0xFF4 3 3 0x00000003
0xFF8 2 2 0x00000002
0xFFC 1 1 0x00000001
Search

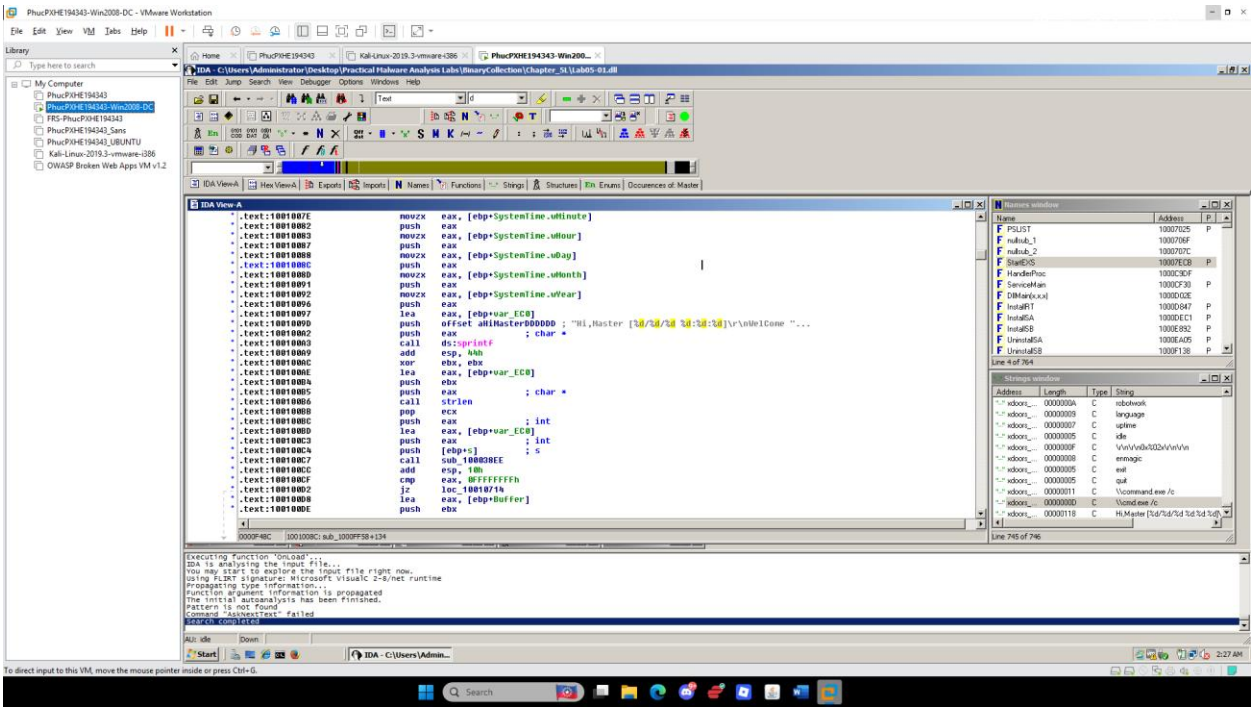

PhucPXHE194343-Win2008-DC - VMware Workstation
IDA - C:\Users\Administrator\Desktop\Practical Malware Analysis Labs\BinaryCollection\Chapter_5L\Lab05-01.dll
Functions window
Updates are not being installed automatically
Click to choose how Windows can install updates.

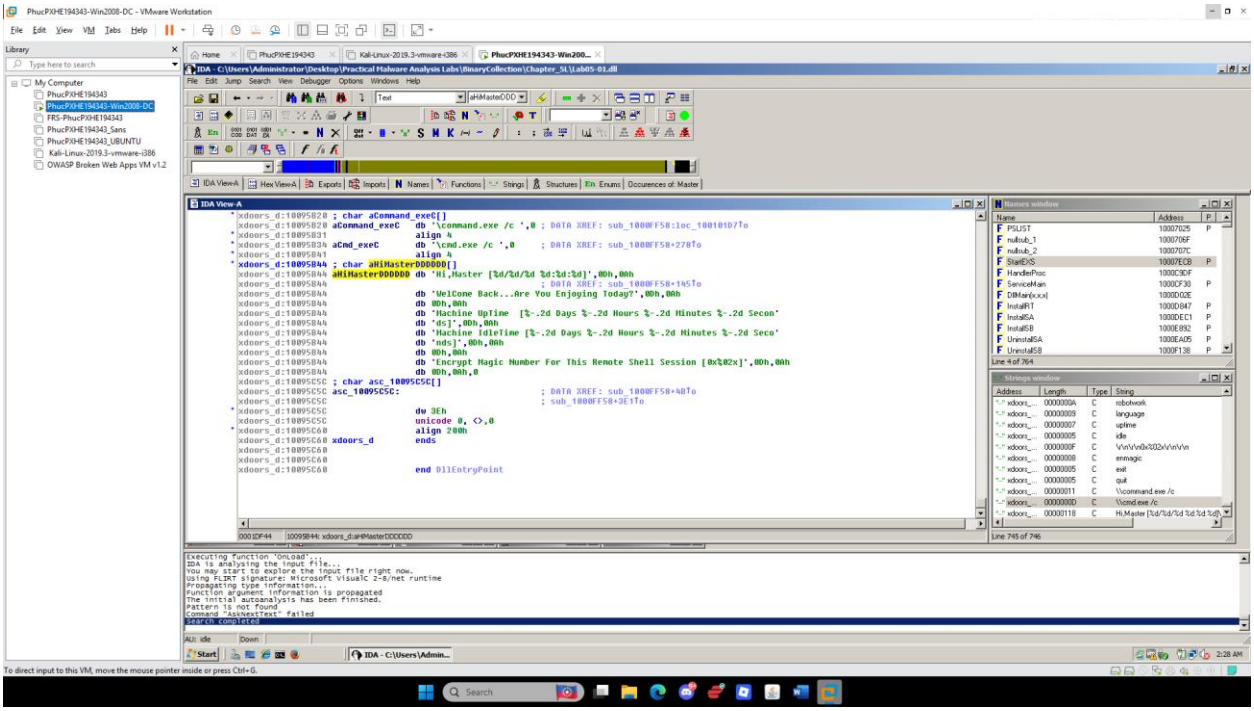

PhucPXHE194343-Win2008-DC - VMware Workstation
IDA - C:\Users\Administrator\Desktop\Practical Malware Analysis Labs\BinaryCollection\Chapter_5L\Lab05-01.dll
Imports


PhucPXHE194343-Win2008-DC - VMware Workstation
IDA - C:\Users\Administrator\Desktop\Practical Malware Analysis Labs\BinaryCollection\Chapter_5L\Lab05-01.dll
.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID)
.text:10001656 sub_10001656 proc near ; DATA XREF: DllMain(x,x,x)+C8↓o
.text:10001656 var_675 = byte ptr -675h
.text:10001656 var_674 = dword ptr -674h
.text:10001656 hModule = dword ptr -670h
.text:10001656 timeout = timeval ptr -66Ch
.text:10001656 name = sockaddr ptr -664h
.text:10001656 var_654 = word ptr -654h
.text:10001656 in = in_addr ptr -650h
.text:10001656 Parameter = byte ptr -64Ah
.text:10001656 CommandLine = byte ptr -63Fh
.text:10001656 Data = byte ptr -638h
.text:10001656 var_544 = dword ptr -544h
.text:10001656 var_50C = dword ptr -50Ch
.text:10001656 var_500 = dword ptr -500h
.text:10001656 var_4FC = dword ptr -4FCh
.text:10001656 readfds = fd_set ptr -4BCh
.text:10001656 phkResult = HKEY__ ptr -388h
.text:10001656 var_388 = dword ptr -388h
.text:10001656 var_1A4 = dword ptr -1A4h
.text:10001656 var_194 = dword ptr -194h
.text:10001656 WSAData = WSAData ptr -190h
.text:10001656 arg_0 = dword ptr 4
.text:10001656 sub esp, 678h
.text:1000165C push ebx
Compiling file 'C:\Program Files\IDA Free\idc\onload.idc'...
Executing function 'OnLoad'...
IDA is analysing the input file...
You may start to explore the input file right now.
Using FLIRT signature: Microsoft VisualC 2-8/net runtime
Propagating type information...
Function argument information is propagated
The initial autoanalysis has been finished.
Retrieving information from the database... ok
Retrieving information from the database... ok

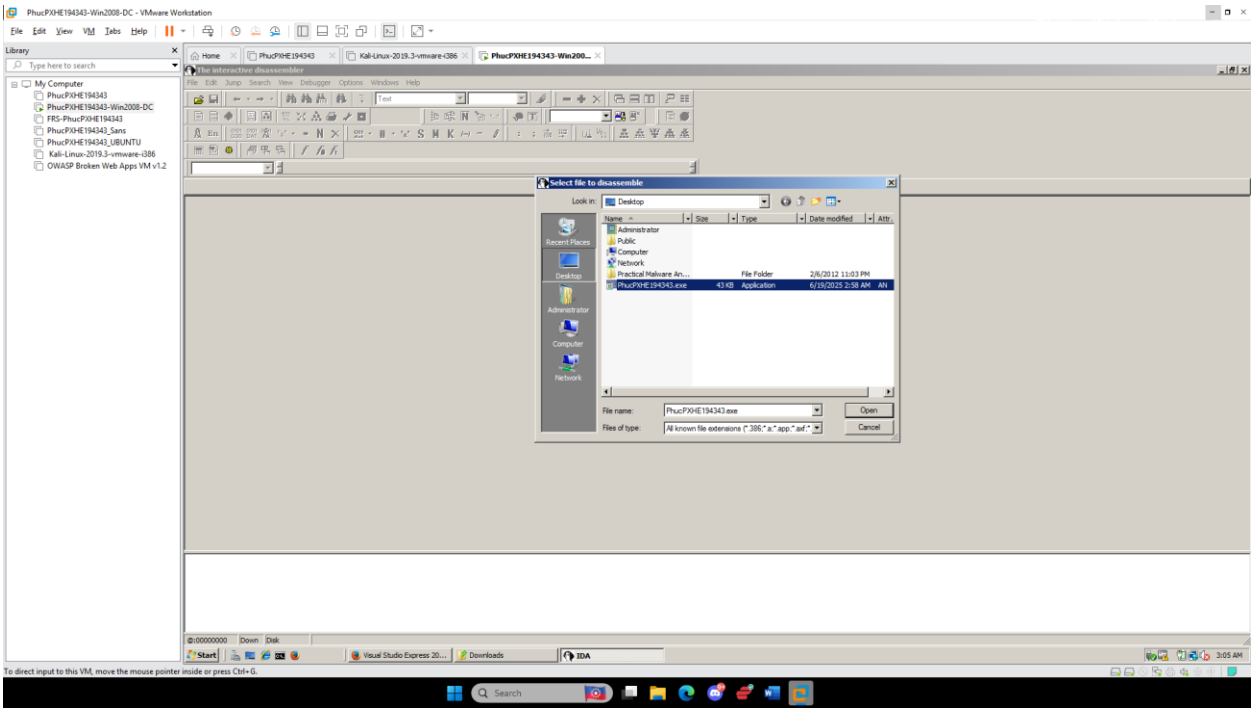

PhucPXHE194343-Win2008-DC - VMware Workstation
IDA - C:\Users\Administrator\Desktop\Practical Malware Analysis Labs\BinaryCollection\Chapter_5L\Lab05-01.dll
xdoors_d:10095B10 aExit db 'exit',0 ; DATA XREF: sub_1000FF58+38D↑o
xdoors_d:10095B15 align 4
xdoors_d:10095B18 ; void aQuit
xdoors_d:10095B18 aQuit db 'quit',0 ; DATA XREF: sub_1000FF58+36F↑o
xdoors_d:10095B1D align 10h
xdoors_d:10095B20 ; char aCommand_exeC[]
xdoors_d:10095B20 aCommand_exeC db '\command.exe /c ',0 ; DATA XREF: sub_1000FF58:loc_10010107↑o
xdoors_d:10095B31 align 4
xdoors_d:10095B34 aCmd_exeC db '\cmd.exe /c ',0 ; DATA XREF: sub_1000FF58+270↑o
xdoors_d:10095B41 align 4
xdoors_d:10095B44 ; char aHiMasterDDDDD[]
xdoors_d:10095B44 aHiMasterDDDDD db 'Hi,Master [%d/%d/%d %d:%d:%d]',0Dh,0Ah
xdoors_d:10095B44 ; DATA XREF: sub_1000FF58+145↑o
xdoors_d:10095B44 db 'WelCome Back...Are You Enjoying Today?',0Dh,0Ah
xdoors_d:10095B44 db 0Dh,0Ah
xdoors_d:10095B44 db 'Machine UpTime [%-.2d Days %-.2d Hours %-.2d Minutes %-.2d Secon'
xdoors_d:10095B44 db 'ds]',0Dh,0Ah
xdoors_d:10095B44 db 'Machine IdleTime [%-.2d Days %-.2d Hours %-.2d Minutes %-.2d Seco'
xdoors_d:10095B44 db 'nds]',0Dh,0Ah
xdoors_d:10095B44 db 0Dh,0Ah
xdoors_d:10095B44 db 'Encrypt Magic Number For This Remote Shell Session [0x%02x]',0Dh,0Ah
xdoors_d:10095B44 db 0Dh,0Ah,0
xdoors_d:10095C5C ; char asc_10095C5C[]
xdoors_d:10095C5C asc_10095C5C: ; DATA XREF: sub_1000FF58+4B7↑o
xdoors_d:10095C5C ; sub_1000FF58+3E1↑o
xdoors_d:10095C5C db 3Eh
xdoors_d:10095C5C unicode 0, <>,0
xdoors_d:10095C60 align 280h
xdoors_d:10095C60 xdoors_d ends
Compiling file 'C:\Program Files\IDA Free\idc\ida.idc'...
Executing function 'main'...
Compiling file 'C:\Program Files\IDA Free\idc\onload.idc'...
Executing function 'OnLoad'...
IDA is analysing the input file...
You may start to explore the input file right now.
Using FLIRT signature: Microsoft VisualC 2-8/net runtime
Propagating type information...
Function argument information is propagated
The initial autoanalysis has been finished.

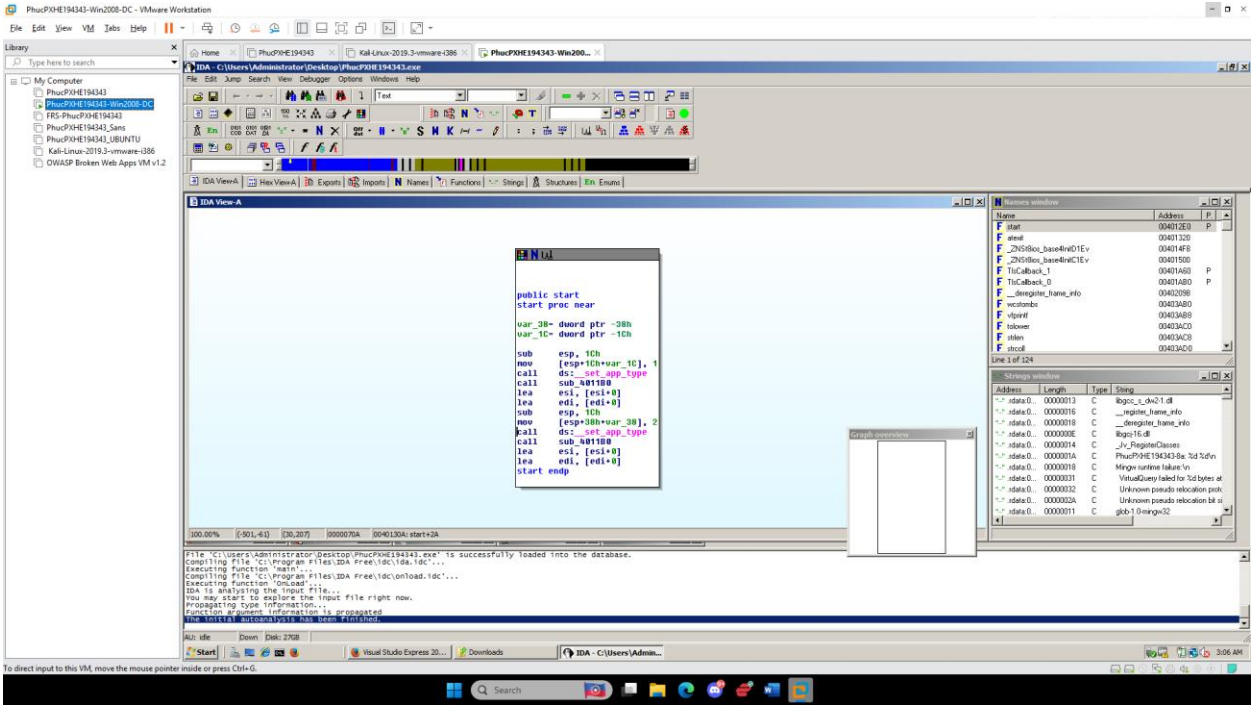


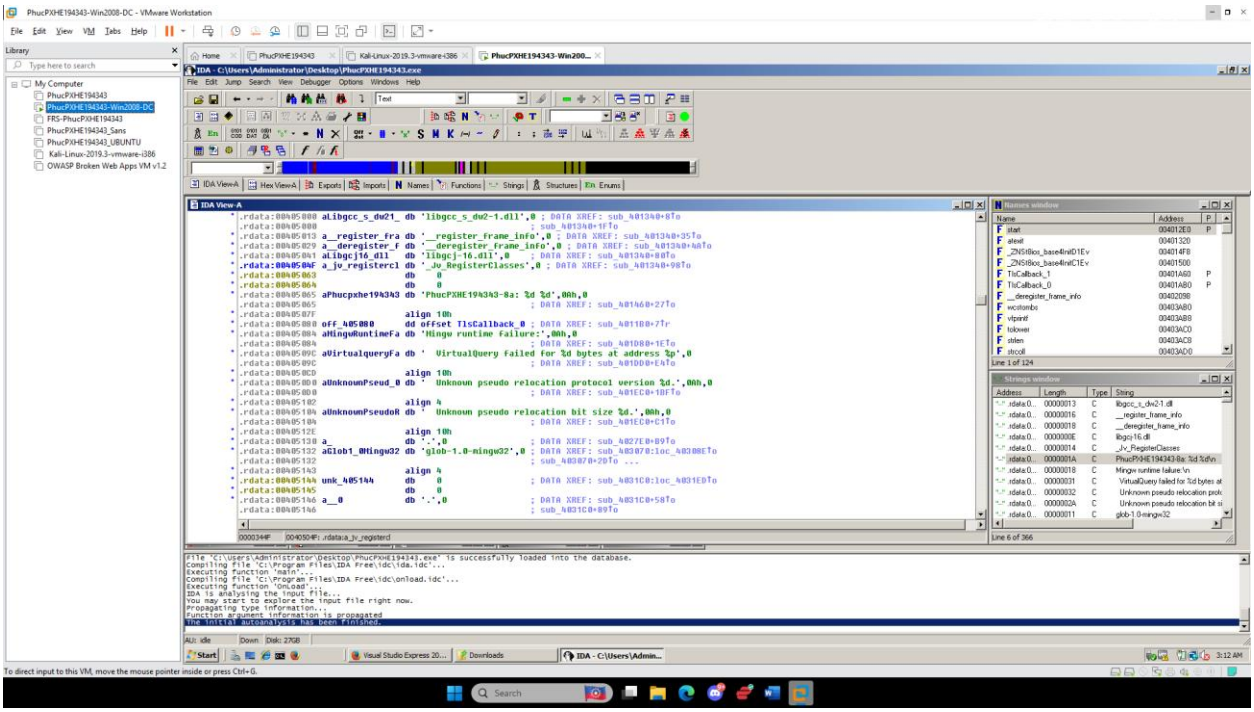


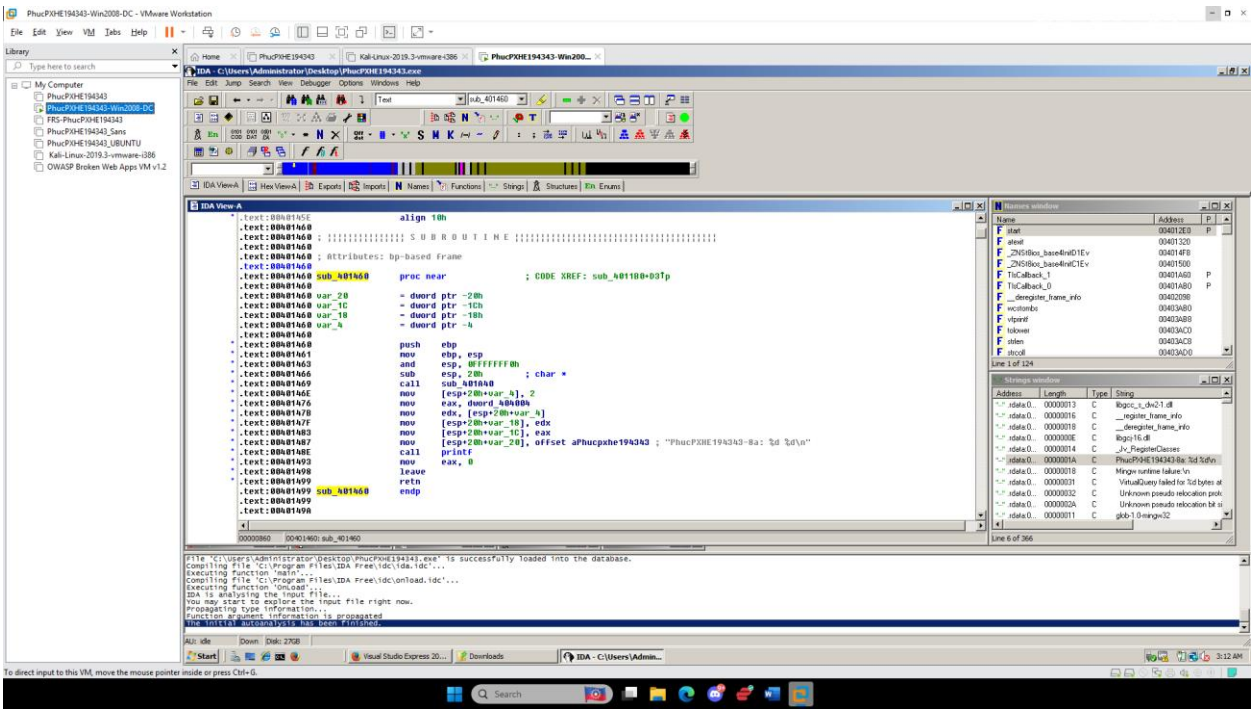


```
PS D:\Documents\IAM\LAB9> cd "d:\Documents\IAM\LAB9\" ; if ($?) { g++ PhucPXHE194343.cpp -o PhucPXHE194343 } ; if ($?) { .\PhucPXHE194343 }
PhucPXHE194343-8a: 1 2
PS D:\Documents\IAM\LAB9>
```



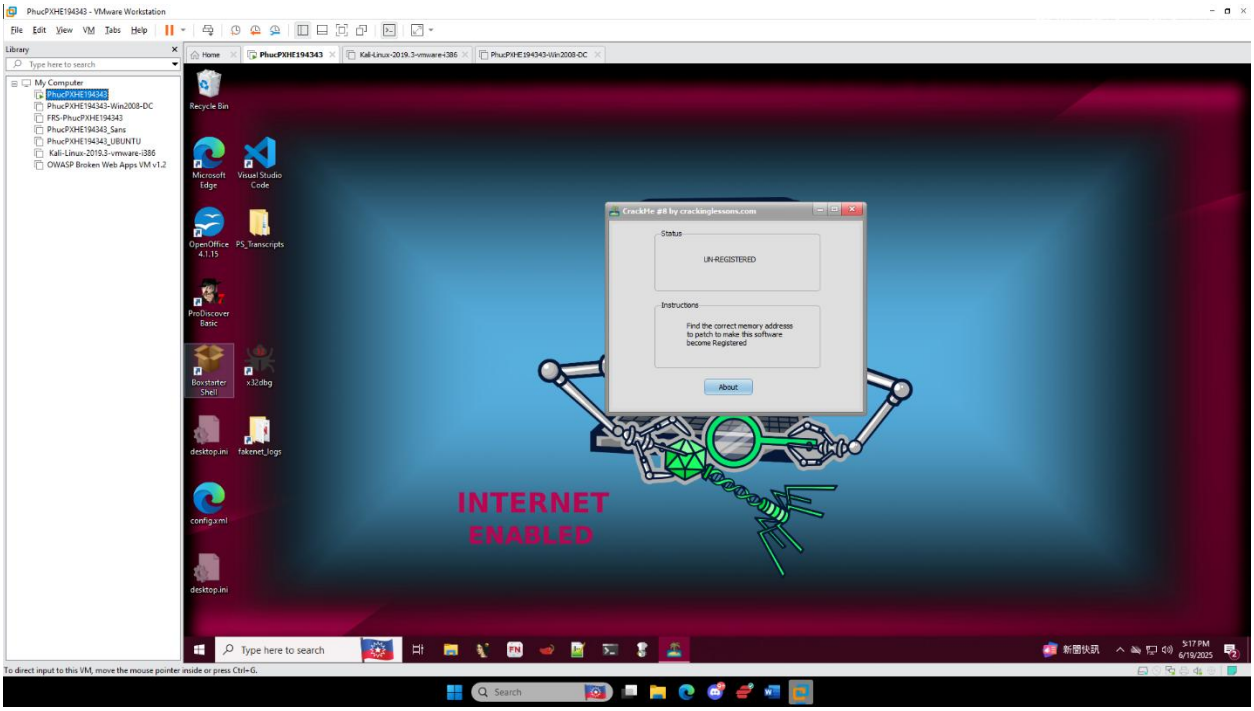




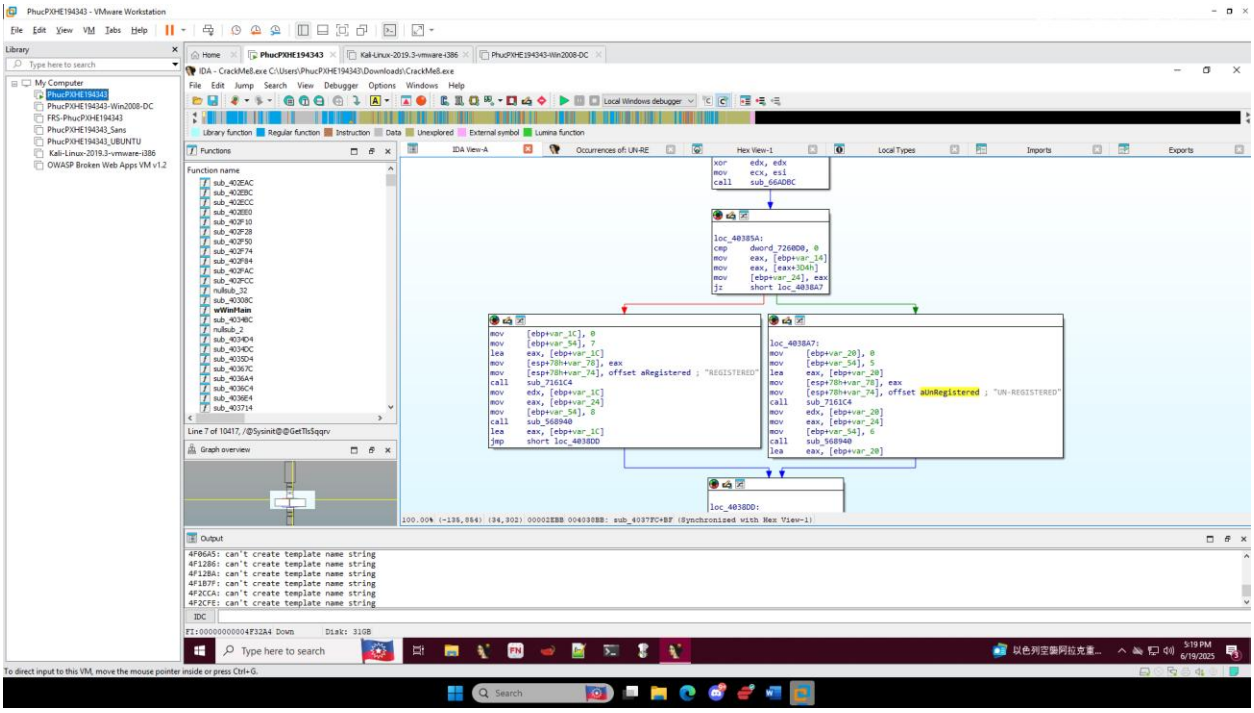


Crack Me 8

Step 1: Open the exe file to check for any requirements



Step 2: Load the Program to IDA for debugging, and search for the string UN-REGISTERED we found an diagram



Step 3: Check registration flag:

cmp dword_7260D0, 0

- If the flag is 0, the program is not registered.
- If it's not 0, it's already registered.

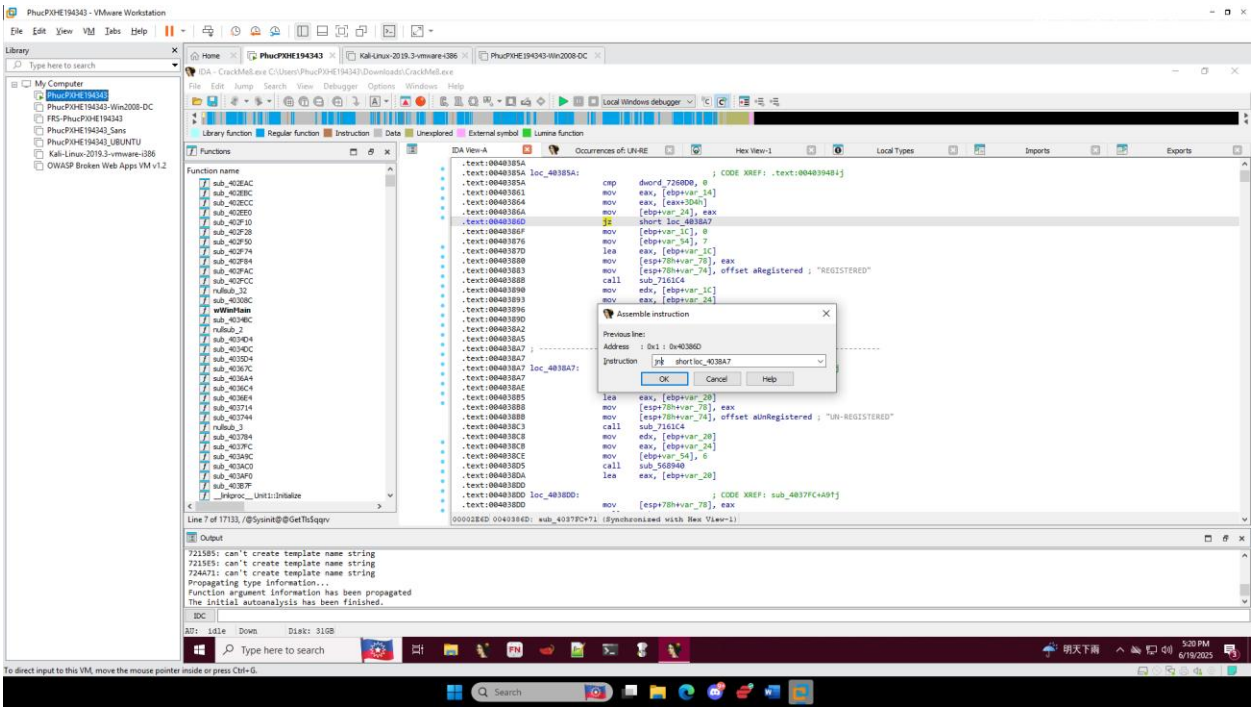
If NOT Registered (dword_7260D0 == 0):

- Prepare the string "REGISTERED".
- Call a function (sub_7161C4) to process that string.
- Pass the result to another function (sub_568940), likely to apply or display the status.

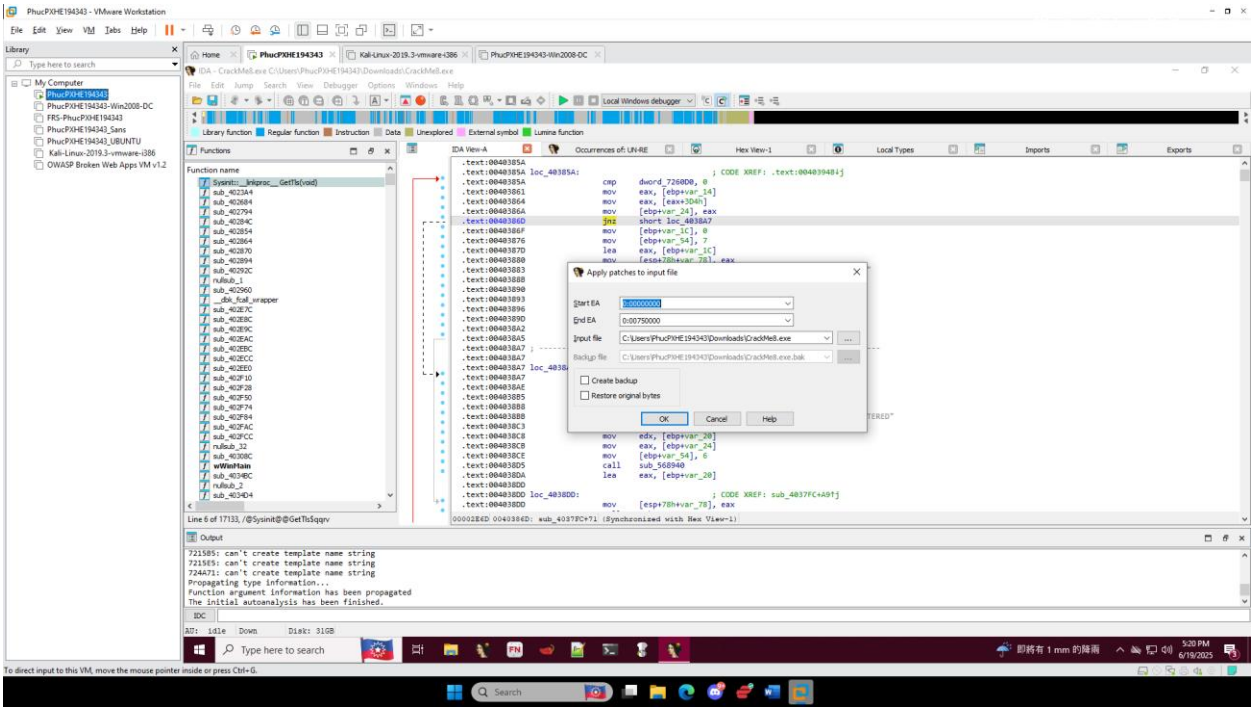
If Registered (dword_7260D0 != 0):

- Prepare the string "UN-REGISTERED".
- Call the same functions with this different label.

So that we change the command jz(jump if zero) to jnz (jump if not zero) which mean if the registration flag is 0 → jump to "UN-REGISTERED" block. If it's not 0 → do "REGISTERED" logic.



Step 4: Apply patches to the program



Step 5: Checking the program is REGISTERED