LAB 6

By: Sebastian Martinez

Code Construct

Q1: Read the introduction of the section 6 "Recognizing C Code Constructs in Assembly" and explain what means a "Code Construct". What aspects may impact the way as assembly code is generated?

R1: Code construct is a code abstraction level that defines a functional property but not the details of its implementation. Also compiler versions and settings can impact how a particular construct appears in disassembly

Q2: Read the section "Global vs Local Variables" and identify what are the differences in the compilation of a code that employs global variables vs one that employs local Variables.

R2: The global variables are referenced by memory addresses, and the local variables are referenced by the stack addresses.

Q3: Read the section "Disassembling Arithmetic Operations" and explain to your classmates how the operations (addition, substraction, increment, decrement and modulo) are represented in assembly code.

R3:

- Addition = add
- Subtraction = sub
- Increment = add,1
- Decrement = sub,1
- Modulo = idiv

Q4: Read the section "Recognizing if Statements" and explain to your classmates how to recognize an if/else structure in assembly code.

R4:

- Comparison = cmp
- Conditional jump = jnz, It’s necessary for the if statement but it’s not always an if statement.
- Jump = jmp

Q5: Read the section "Recognizing Nested if Statements" and explain to your classmates how to recognize a "Nested IF" structure in assembly code.

R5: Three different conditional jumps occur. The first occurs if var_4 does not equal var_8. The other two occur if var_C is not equal to zero.

Q6: Read the section "Recognizing Loops" and explain to your classmates how to recognize a FOR structure in assembly code.

R6: In assembly, the for loop can be recognized by locating the four components—initialization, comparison, execution instructions, and increment/ decrement.

Q7: Read the section "Recognizing Loops" and explain to your classmates how to recognize a WHILE structure in assembly code.

R7: It looks similar to the for loop, but the only way for this code to stop executing repeatedly is for that conditional jump to occur.

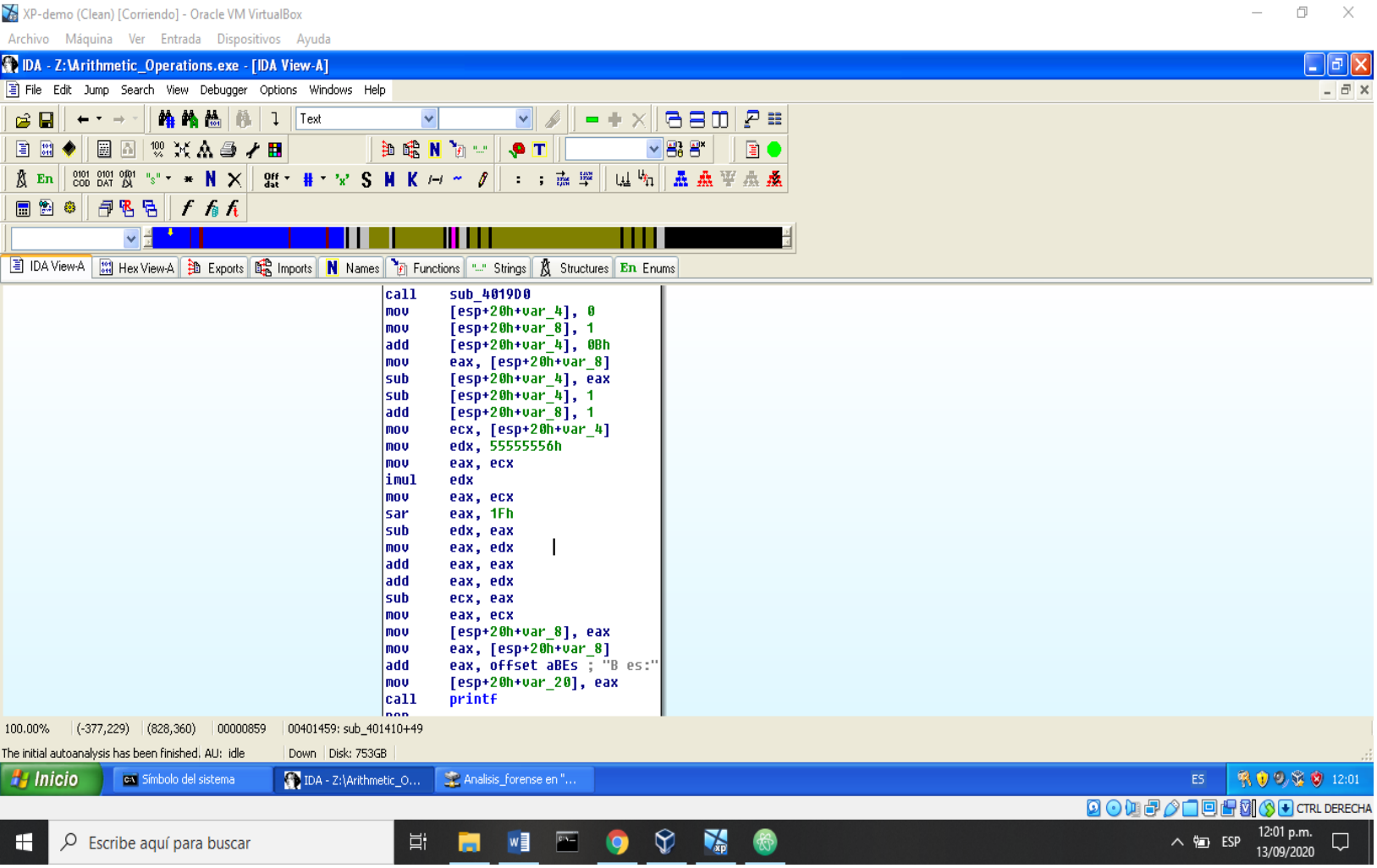
Q8: Read the section "Understanding Function Call Convenstions" and explain to your classmates how to recognize a "function call" in assembly code.

R8:

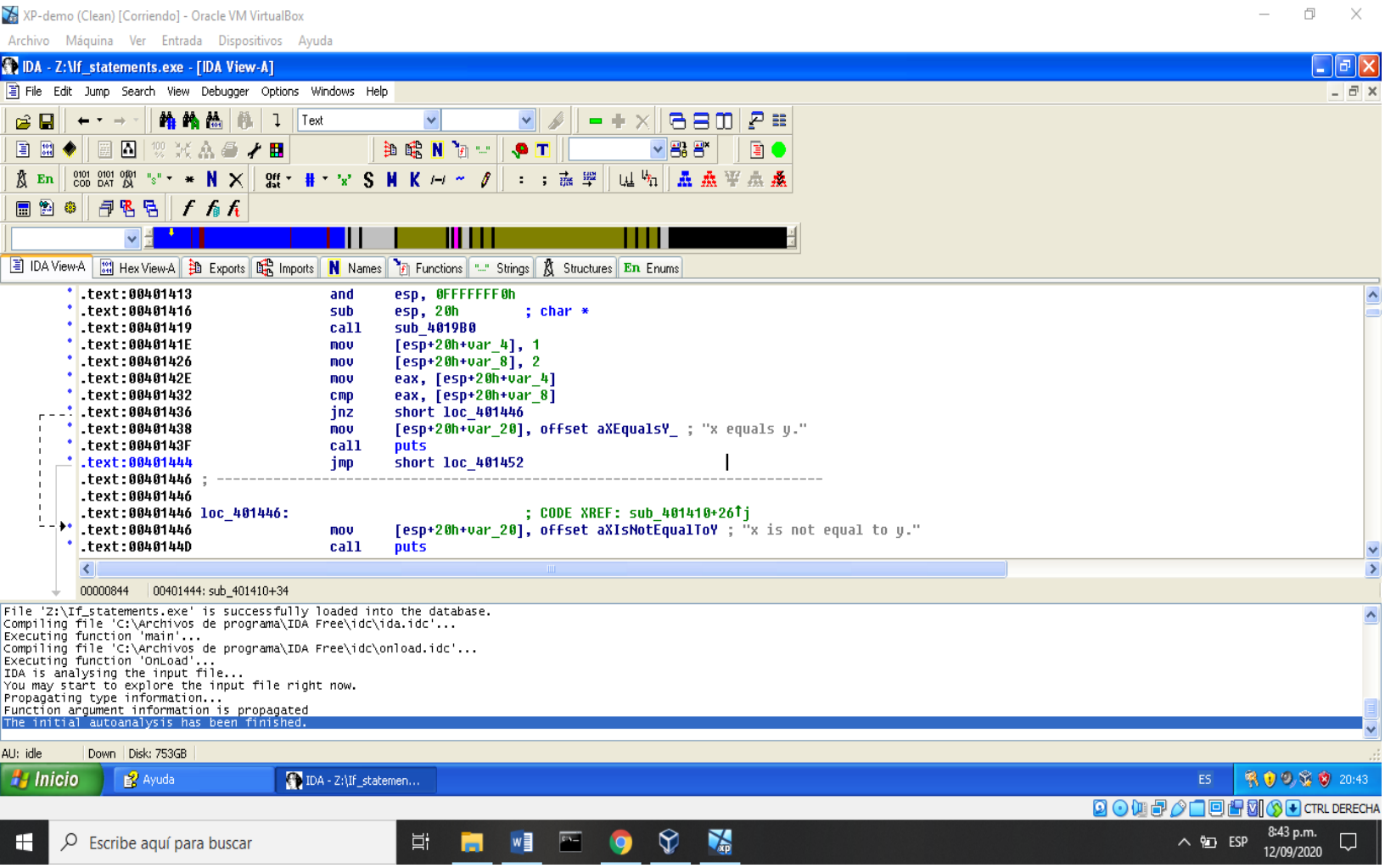
Q9: Read the section "Analyzing switch Statements" and explain to your classmates how to recognize a switch structure in assembly code.

R9:

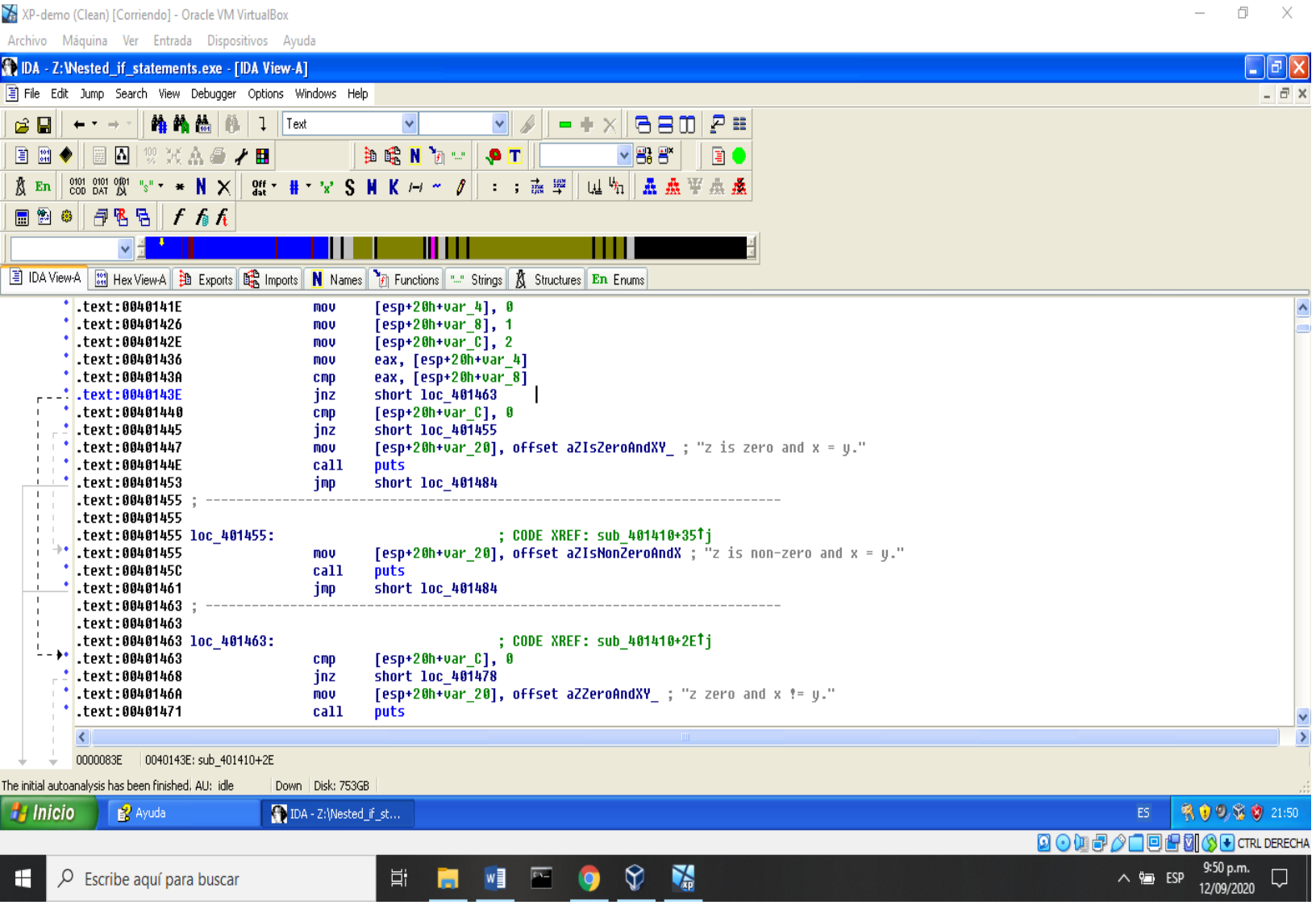
ARITHMETIC OPERATIONS



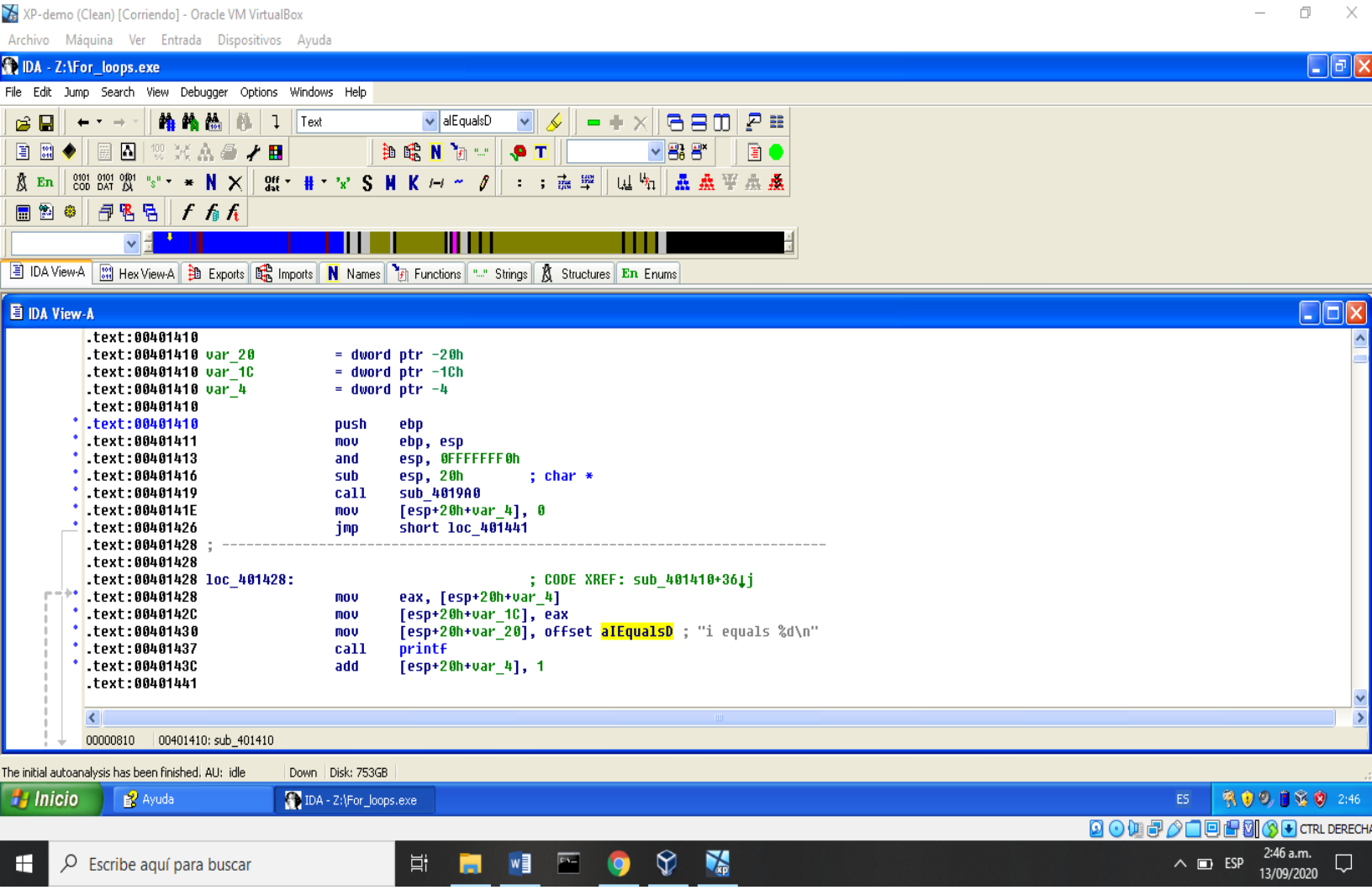
IF STATEMENTS



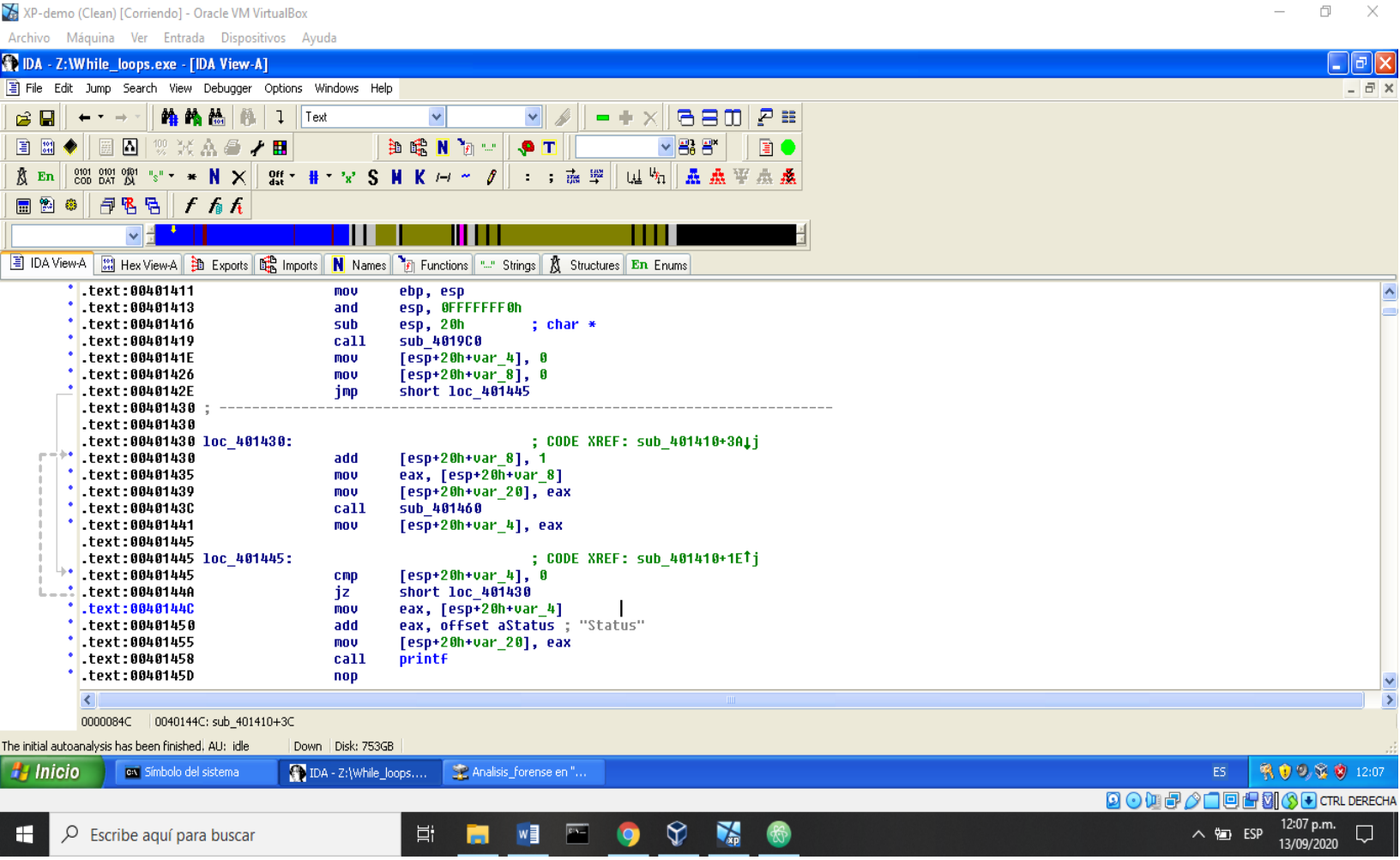
NESTED IF STATEMENTS



FOR LOOPS


XP-demo (Clean) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
IDA - Z:\For_loops.exe
File Edit Jump Search View Debugger Options Windows Help
IDA View-A
Hex View-A
Exports
Imports
Names
Functions
Strings
Structures
Enums
.text:00401410
.text:00401410 var_20 = dword ptr -20h
.text:00401410 var_1C = dword ptr -1Ch
.text:00401410 var_4 = dword ptr -4
.text:00401410
.text:00401410 push ebp
.text:00401411 mov ebp, esp
.text:00401413 and esp, 0FFFFFFF0h
.text:00401416 sub esp, 20h ; char *
.text:00401419 call sub_4019A0
.text:0040141E mov [esp+20h+var_4], 0
.text:00401426 jmp short loc_401441
.text:00401428 ; ---
.text:00401428
.text:00401428 loc_401428: ; CODE XREF: sub_401410+36↓j
.text:00401428 mov eax, [esp+20h+var_4]
.text:0040142C mov [esp+20h+var_1C], eax
.text:00401430 mov [esp+20h+var_20], offset aIEqualsD ; "i equals %d\n"
.text:00401437 call printf
.text:0040143C add [esp+20h+var_4], 1
.text:00401441
00000810 00401410: sub_401410
The initial autoanalysis has been finished. AU: idle Down Disk: 753GB
Inicio
Ayuda
IDA - Z:\For_loops.exe
ES
2:46
CTRL DERECHA
Escribe aquí para buscar
ESP
2:46 a.m.
13/09/2020

WHILE LOOPS


XP-demo (Clean) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
IDA - Z:\While_loops.exe - [IDA View-A]
File Edit Jump Search View Debugger Options Windows Help
IDA View-A Hex View-A Exports Imports Names Functions Strings Structures Enums
.text:00401411 mov ebp, esp
.text:00401413 and esp, 0FFFFFFF0h
.text:00401416 sub esp, 20h ; char *
.text:00401419 call sub_4019C0
.text:0040141E mov [esp+20h+var_4], 0
.text:00401426 mov [esp+20h+var_8], 0
.text:0040142E jmp short loc_401445
.text:00401430 ; ---------- (----------)
.text:00401430
.text:00401430 loc_401430: ; CODE XREF: sub_401410+3A↓j
.text:00401430 add [esp+20h+var_8], 1
.text:00401435 mov eax, [esp+20h+var_8]
.text:00401439 mov [esp+20h+var_20], eax
.text:0040143C call sub_401460
.text:00401441 mov [esp+20h+var_4], eax
.text:00401445
.text:00401445 loc_401445: ; CODE XREF: sub_401410+1E↑j
.text:00401445 cmp [esp+20h+var_4], 0
.text:0040144A jz short loc_401430
.text:0040144C mov eax, [esp+20h+var_4]
.text:00401450 add eax, offset aStatus ; "Status"
.text:00401455 mov [esp+20h+var_20], eax
.text:00401458 call printf
.text:0040145D nop
0000084C 0040144C: sub_401410+3C
The initial autoanalysis has been finished. AU: idle Down Disk: 753GB
Inicio
Símbolo del sistema
IDA - Z:\While_loops....
Analisis_forense en "...
ES
12:07
CTRL DERECHA
Escribe aquí para buscar
ESP
12:07 p.m.
13/09/2020