

# Progetto W10

Analisi Malware e Codice Assembly

# Traccia

Con riferimento al file `Malware_U3_W2_L5` presente all'interno della cartella

«Esercizio\_Pratico\_U3\_W2\_L5» sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

Quali librerie vengono importate dal file eseguibile?

Quali sono le sezioni di cui si compone il file eseguibile del malware?

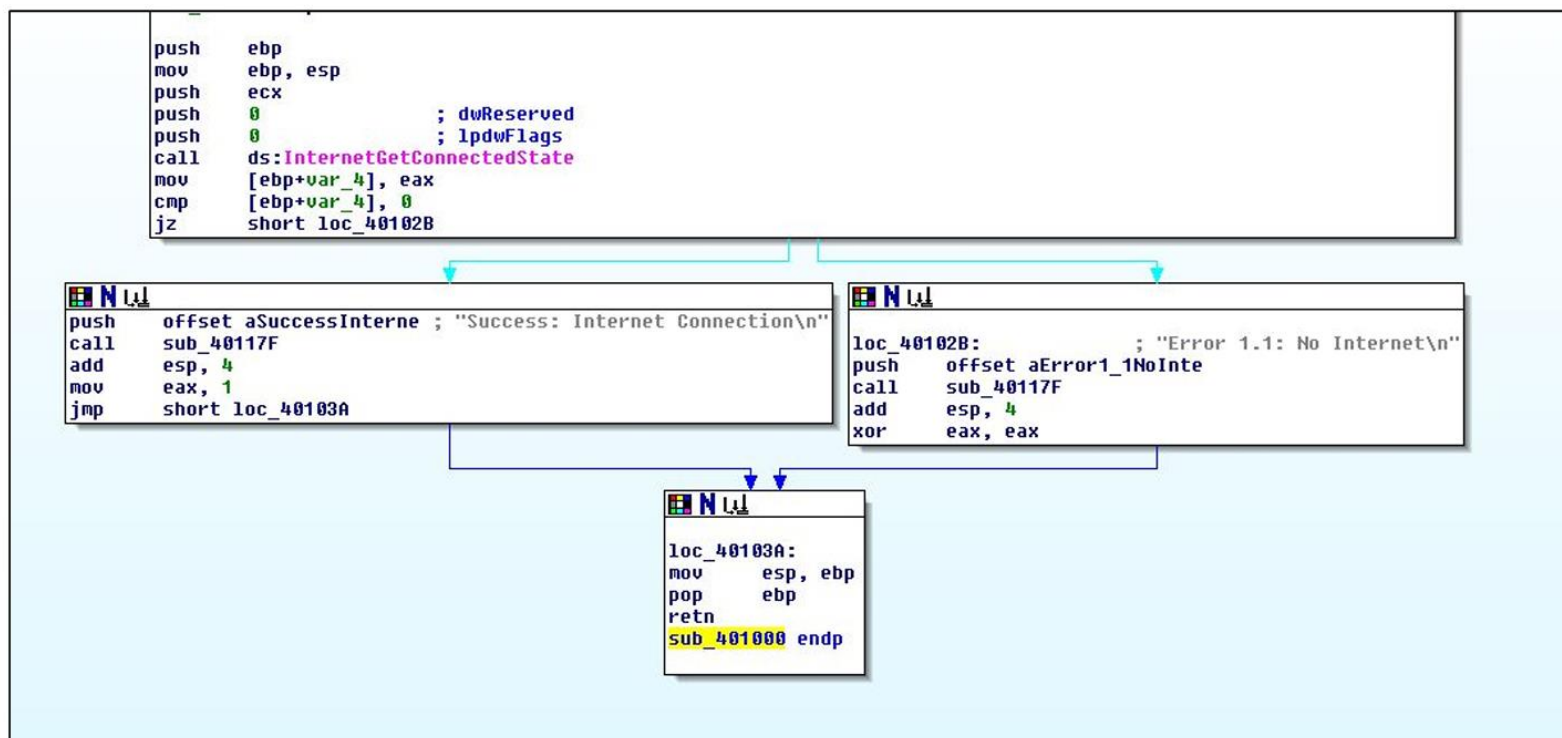
Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)

Ipotizzare il comportamento della funzionalità implementata

# Traccia

Figura 1

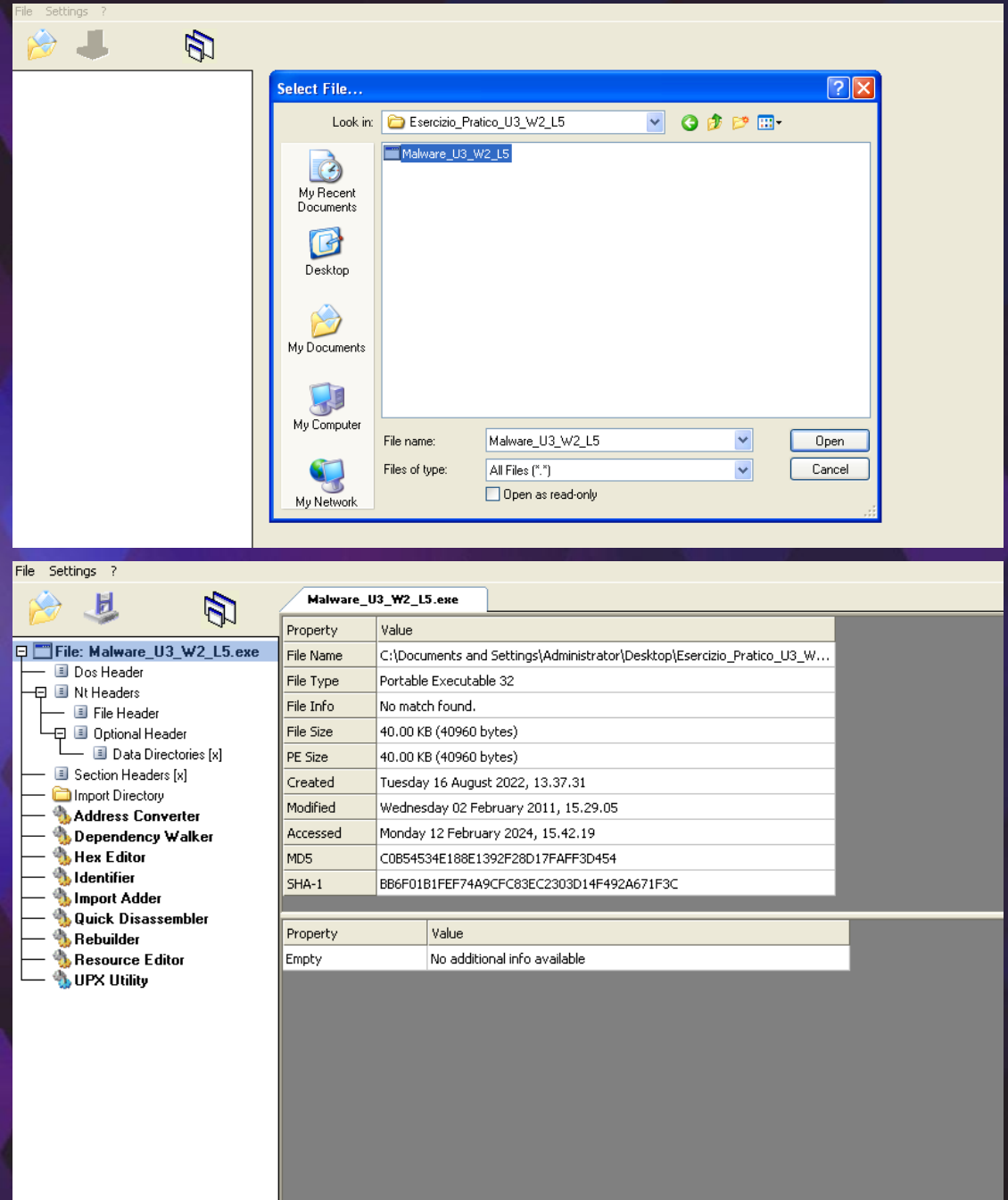


# Inizio prima parte

Analisi malware

# Utilizzo di CFF Explorer

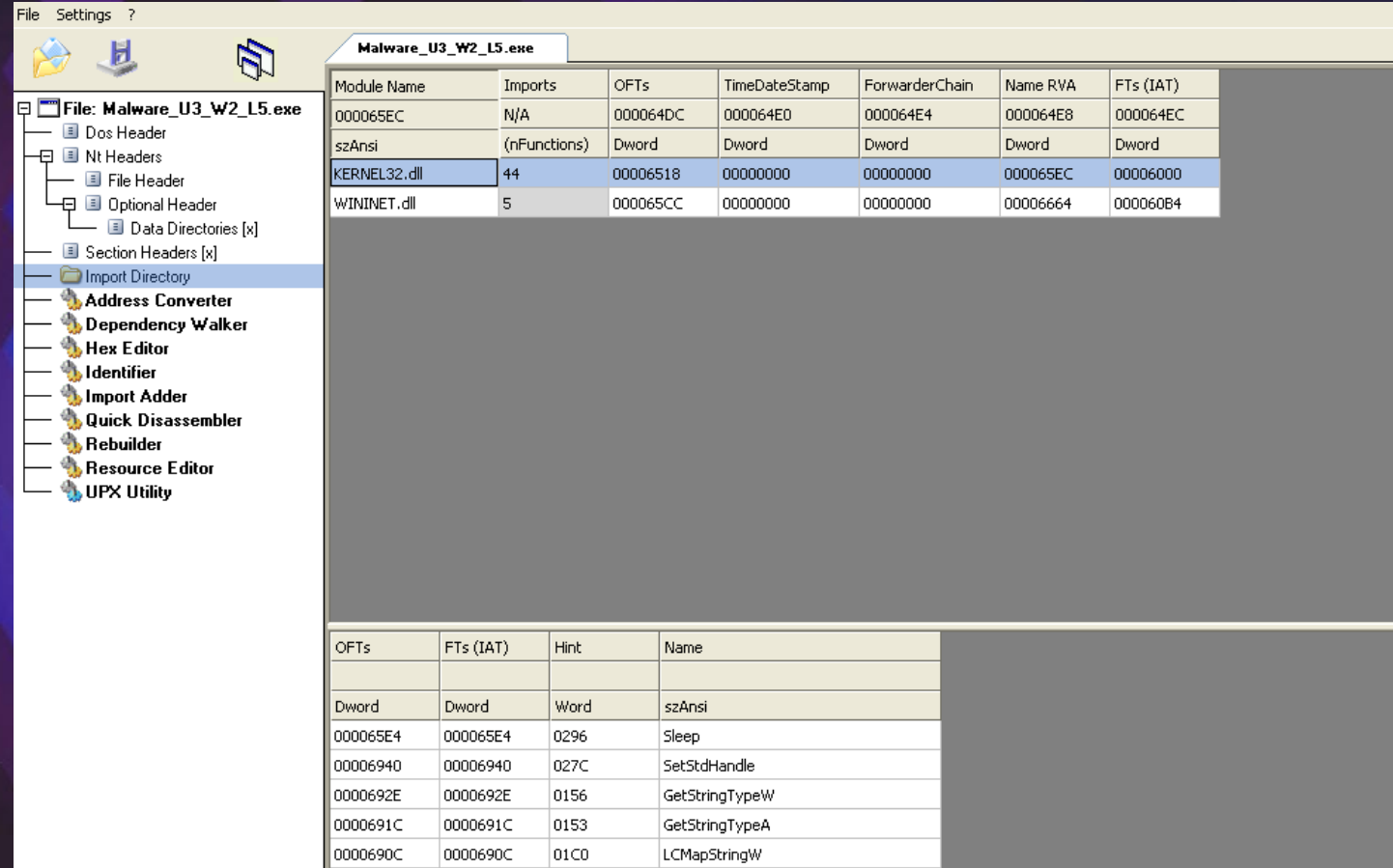
Dopo l'avvio di CFF Explorer, un'applicazione già presente nella configurazione della macchina, è stato aperto il file indicato nella traccia. CFF Explorer è uno strumento avanzato per l'analisi di file eseguibili, noto per la sua capacità di esaminare a fondo le funzioni importate ed esportate di un malware. Questa caratteristica consente di comprendere meglio il comportamento del malware e identificare eventuali minacce o vulnerabilità nel sistema.





# Analisi Import Directory

Selezionando la directory di importazione nella cartella, è stato agevole esaminare l'elenco delle librerie importate dal malware. La tabella sottostante fornisce un'analisi dettagliata della libreria attualmente evidenziata, come nel caso della KERNEL32.DLL, la quale racchiude le funzioni fondamentali per interagire con il sistema operativo, come la manipolazione dei file e la gestione della memoria. WININET.dll è una libreria di sistema di Windows che consente alle applicazioni di connettersi e comunicare su Internet.



The screenshot shows the 'Import Directory' window in Immunity Debugger. The left pane displays the file structure of 'Malware\_U3\_W2\_L5.exe', with 'Import Directory' selected. The main pane shows a table of imported modules and functions.

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
000065EC	N/A	000064DC	000064E0	000064E4	000064E8	000064EC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep
00006940	00006940	027C	SetStdHandle
0000692E	0000692E	0156	GetStringTypeW
0000691C	0000691C	0153	GetStringTypeA
0000690C	0000690C	01C0	LCMapStringW

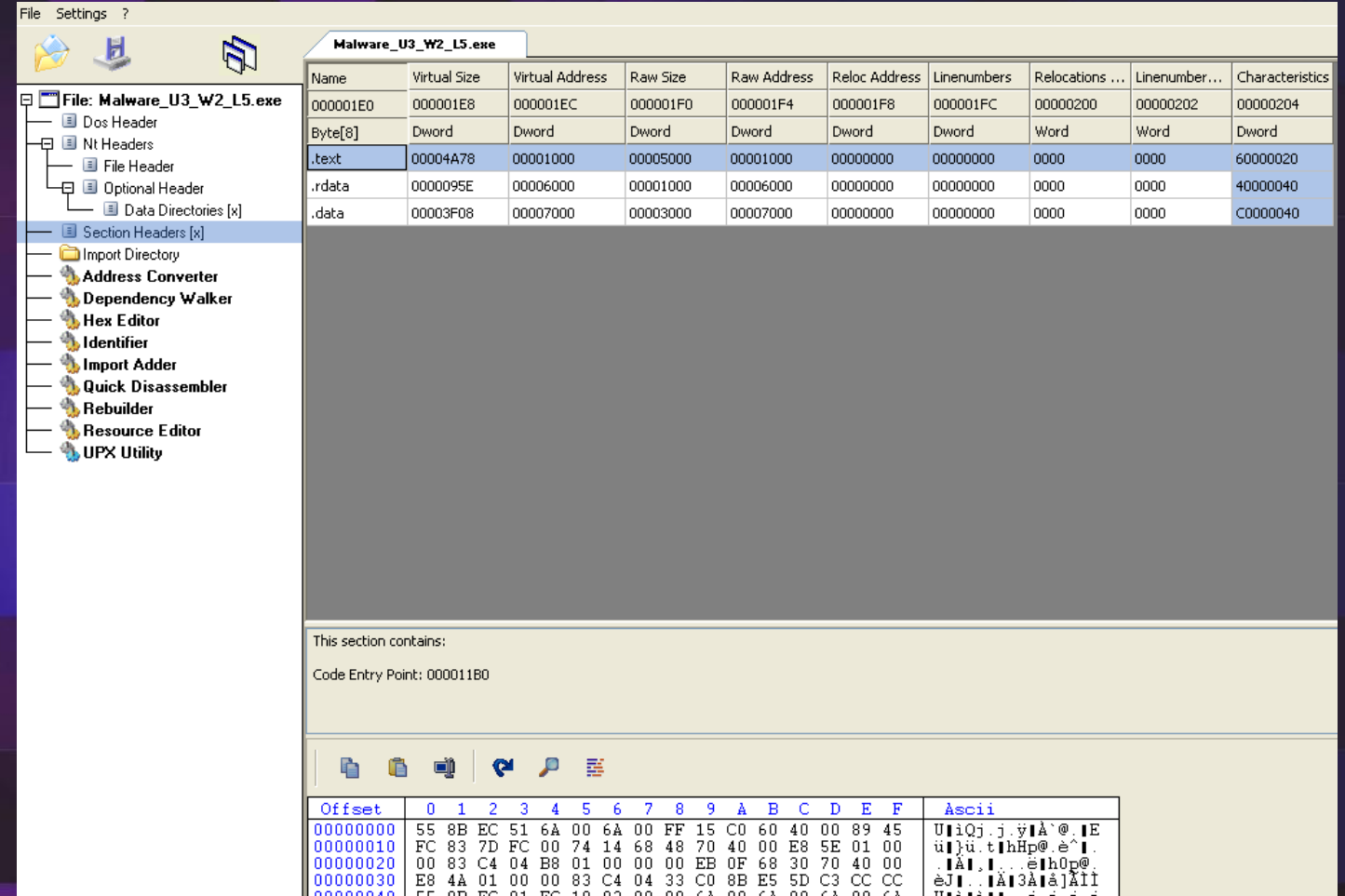
# Analisi Section Headers

Nella sezione Section Headers, è stato possibile esaminare le informazioni relative alle sezioni costituenti dell'eseguibile. Il malware è suddiviso in tre parti:

**.text:** Contiene il codice eseguibile del malware.

**.rdata:** Contiene dati di sola lettura, come stringhe di testo e costanti.

**.data:** Contiene dati modificabili durante l'esecuzione del malware, come variabili e configurazioni.



File: Malware\_U3\_W2\_L5.exe

Section Headers [x]

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
000001E0	000001E8	000001EC	000001F0	000001F4	000001F8	000001FC	00000200	00000202	00000204
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

This section contains:

Code Entry Point: 000011B0

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	55	8B	EC	51	6A	00	6A	00	FF	15	C0	60	40	00	89	45	UtiQj.j.y A'@.IE
00000010	FC	83	7D	FC	00	74	14	68	48	70	40	00	E8	5E	01	00	u}u.t hHp@.è^l.
00000020	00	83	C4	04	B8	01	00	00	00	EB	0F	68	30	70	40	00	A ...è h0p@.
00000030	E8	4A	01	00	00	83	C4	04	33	C0	8B	E5	5D	C3	CC	CC	èJ ... A 3A è A l
00000040	55	8B	EC	51	6A	00	6A	00	FF	15	C0	60	40	00	89	45	UtiQj.j.y A'@.IE

# Inizio seconda parte

Analisi Codice Assembly



## Costrutto stack

Queste istruzioni iniziano la creazione di un "frame di stack"

push ebp: Salva il valore corrente di EBP nello stack.

mov ebp, esp: Imposta EBP al valore corrente di ESP, creando un nuovo frame di stack.

In breve, queste istruzioni creano un nuovo frame di stack per la funzione, permettendo l'accesso a parametri e variabili locali usando offset rispetto a EBP.

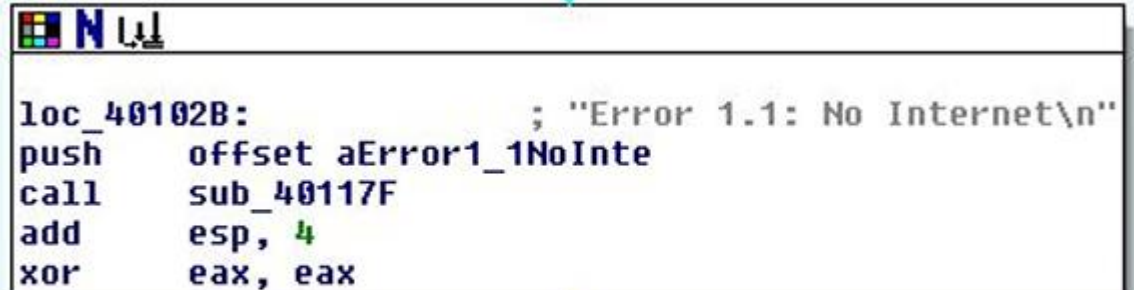
```
push    ebp
mov     ebp, esp
push    ecx
push    0           ; dwReserved
push    0           ; lpdwFlags
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

## Costrutti vari

Successivamente si possono notare varie operazioni come `add esp, 4` che si occupa di aggiungere il valore 4 al registro dello stack pointer (ESP), incrementando così il puntatore dello stack di 4 byte. Xor in assembly esegue un'operazione logica bit a bit tra due operandi. Il risultato è 1 se i bit corrispondenti sono diversi, altrimenti è 0.



```
push    offset aSuccessInterne ; "Success: Inte
call    sub_40117F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```



```
loc_40102B:                                ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
add     esp, 4
xor     eax, eax
```

## Costrutti vari

`mov esp, ebp`: Copia il valore corrente del registro di base (EBP) nello stack pointer (ESP), sovrascrivendo il valore di ESP con il valore di EBP. Questo cambia il puntatore dello stack per puntare alla base del frame di stack corrente.

`pop ebp`: Estrae il valore dalla cima dello stack e lo memorizza nel registro di base (EBP), ripristinando così il valore di EBP a quello precedente al frame di stack corrente.

In sintesi, queste istruzioni ripristinano il frame di stack precedente, permettendo al programma di tornare al contesto della funzione chiamante.



The screenshot shows a snippet of assembly code. At the top, there is a header with a logo and the text "N 4.1". Below this, the code is as follows:

```
loc_40103A:  
mov     esp, ebp  
pop     ebp  
retn  
sub_401000 endp
```

A red rectangular box highlights the instructions `mov esp, ebp` and `pop ebp`. The instruction `sub_401000 endp` is highlighted in yellow.



Fine della presentazione

Amedeo Natalizi