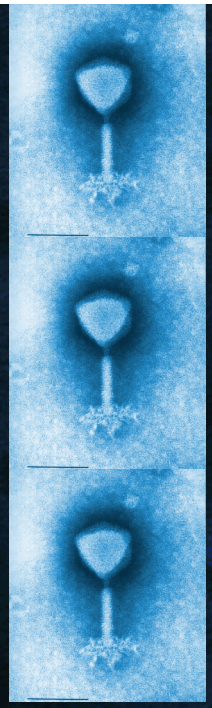


Analyzing Memory Accesses in Obfuscated x86 Executables

Michael Venable
Mohamed R. Choucane
Md. Enamul Karim
Arun Lakhotia (Presenter)

DIMVA 2005
Wien

Talk Contents

- Motivation
- Previous work
  - VSA (Balakrishnan and Reps, 2004)
  - Abstract Stack Graph (Lakhotia and Kumar 2004)
- VSA+ASG
  - Domain
  - Interpretation example
- Application
  - Detect Call obfuscations
- Implementation
- Future and Conclusions

Motivation

- Past project: detecting malicious behavior
  - Match models of malicious behavior
    - models typically involved *system calls*
- *Typical application of accepted techniques*
  - implemented prototype using IDA Pro
  - standard application of formal theory (model checking)
  - tried against common virus (Win32.Evol)

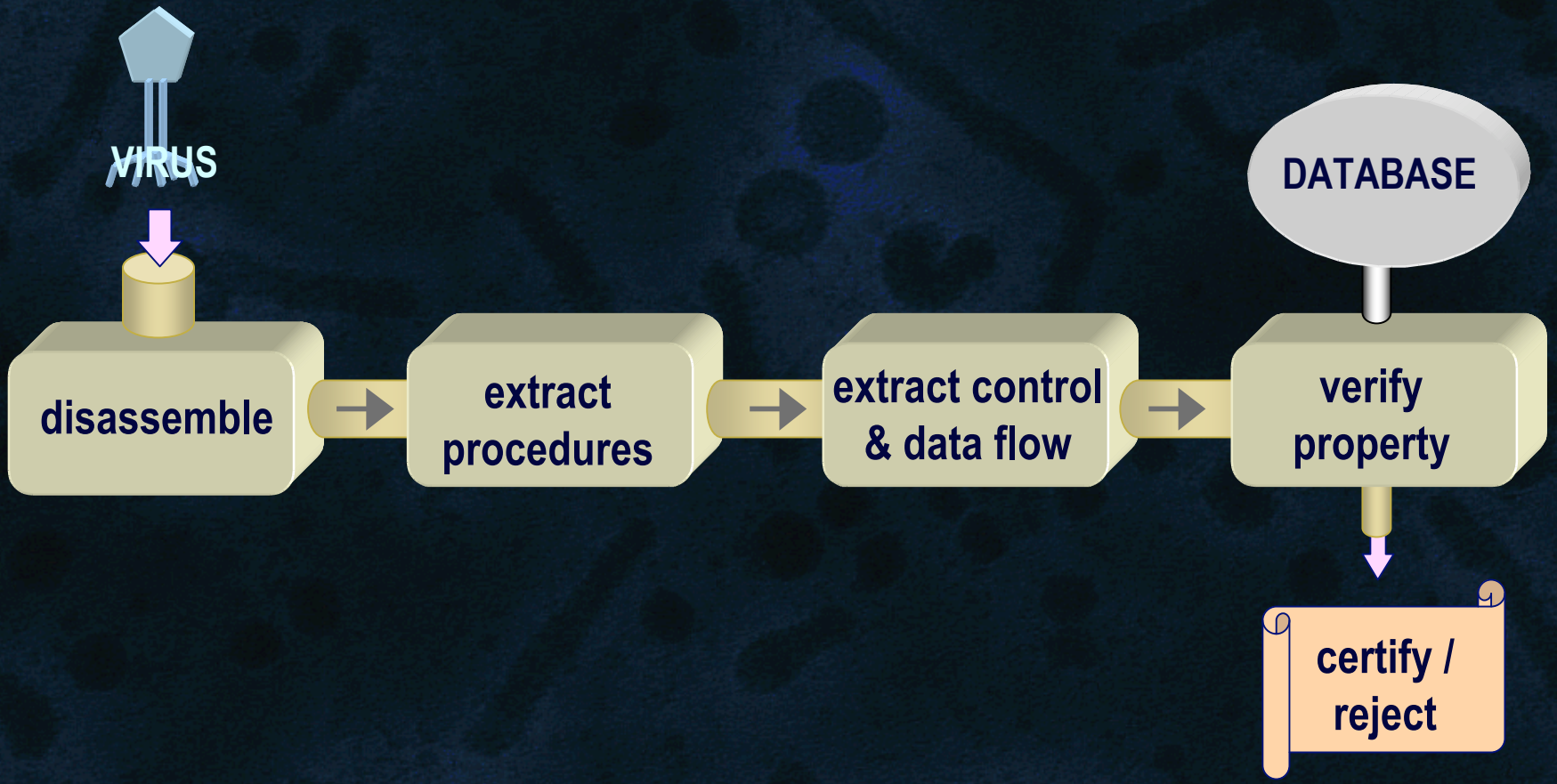
Result: FAILURE

- Our original tool didn't stand a chance
  - our technology wasn't even tested
  - virus writer fouled up the *entire analysis pipeline*
- Success out of failure:
  - showed us where a major battle lies
  - refocus on hardening analysis techniques

Code analysis: benign/adversarial

- Half a century of program analysis
  - compilers, optimizers, checkers, refactoring tools
  - read in programs, analyze, visualize, transform
- Mostly for *friendly* programs
  - yet these are often used to battle malware
  - not prepared to face adversarial code
    - **soft targets**
    - **fragile infrastructures**
  - would we send Mouseketeers to battlefield?

Typical analysis pipelines



Problem: Not hardened

Attack: Disassembly

	ORIG BYTES	ASSEMBLY
401063:	5d	pop %ebp
401064:	c3	ret
401065:	55	push %ebp
401066:	89 e5	mov %esp,%ebp
401068:	83 ec 08	sub $0x8,%esp
40106b:	eb 05	jmp 0x401072
40106d:	e8 ee ff ff ff e8	mov $0x[illegible],%esi
401073:	e9 [illegible]	jmp 0x[illegible]
401078:	45 fc 00 00 00 00	mov %eax,0xfffffffc(%ebp)
40107e:	81 7d fc e7 03 00 00	cmpl $0x3e7,0xfffffffc(%ebp)

bad disassembly (no jump target)

jump over junk

malicious func

Attack: Extract procedures

determine procedure boundaries (entry & exit)

```
401063:    5d                   pop   %ebp
401064:    c3                   ret
401065 <_malicious>:
401065:    55                   push  %ebp
401066:    89 e5                mov   %esp,%ebp
401068:    83 ec 08             sub   $0x8,%esp
40106b:    eb 05                jmp   401072 <_malicious+0xd>
40106d:    e8 ee ff ff ff       call  401060 <_sendLotsOfEmail>
401072:    e8 e9 ff ff ff       call  401060 <_sendLotsOfEmail>
401077:    c7 45 fc 00 00 00 00 movl  $0x0,0xfffffffc(%ebp)
```

malicious func

Attack: Extract CF & DF

trace call structure (control flow)

```
401063:    5d                      pop   %eb
401064:    c3                      ret
401065 <_malicious>:
401065:    55                      push  %ebp
401066:    89 e5                   mov   %esp,%ebp
401068:    83 ec 08                sub   $0x8,%esp
40106b:    eb 35                   jmp   401078 <_malicious+0x13>
40106d:    ff 35 6f 10 40 00       push  0x40106f
401073:    e8 e9 ff ff ff          call  401060 <_sendLotsOfEmail>
401078:    c7 45 fc 00 00 00 00    movl  $0x0,0xfffffffc(%ebp)
```

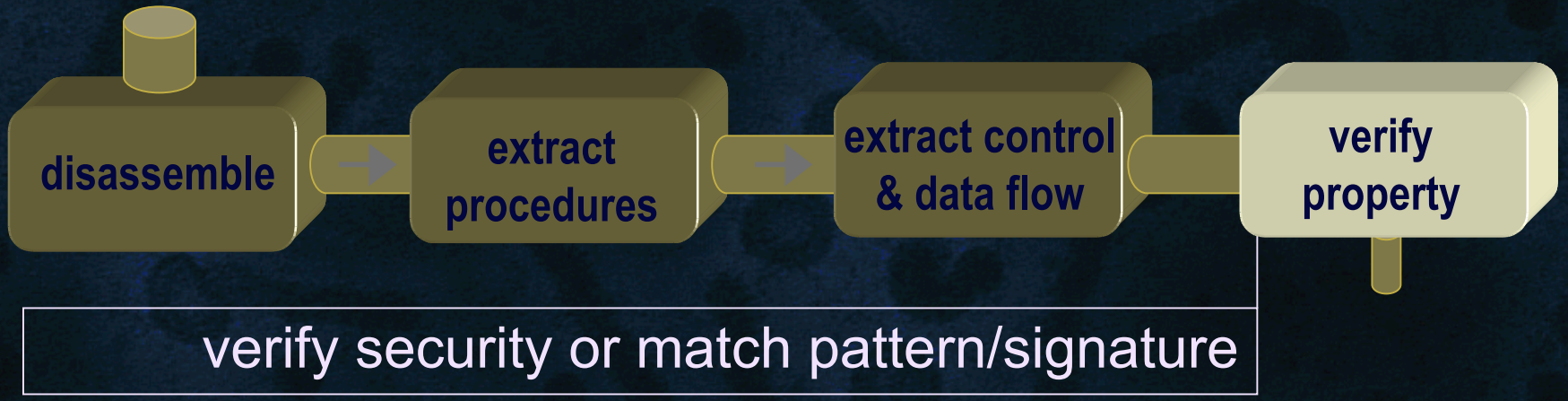
L0: call F
L1: →

L0: push L1
push F
L1: ret

instr. substitution

no call found

Attack: Verify property




```
push x          push x
push y   →      push z
ret             pop
                push y
                ret
```

```
pushl  401078 <_malicious+0x13>
pushl  401060 <_sendLotsOfEmail>
ret
movl   $0x0,0xfffffffc(%ebp)
```

- Transformations destroy signature/pattern match
  - eg metamorphic viruses: self-transforming
  - instruction substitution, nop insertion, etc.

Motivation summary

- Soft target attacks
  - disassembler disruption
  - obfuscated calls
  - metamorphic transformations
- Pipeline disruptions
  - silent failures from every stage

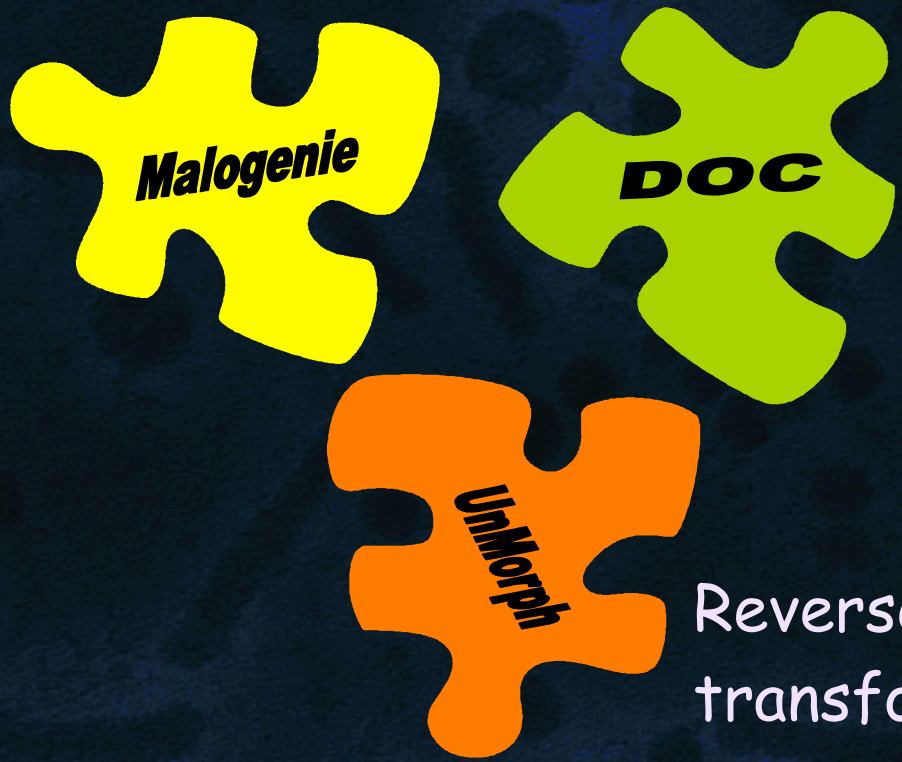
A shift in research focus

Question

How to make program analysis battle-ready?

Advertisement: Portfolio of results

Create malware phylogeny



Deobfuscate Calls

Reverse self transformations

All results are “Patent pending”

Talk Contents

- Motivation
- Previous work
  - VSA (Balakrishnan and Reps, 2004)
  - Abstract Stack Graph (Lakhotia and Kumar 2004)
- VSA+ASG
  - Domain
  - Interpretation example
- Application
  - Detect Call obfuscations
- Implementation
- Future and Conclusions

Value-Set Analysis

- Determines values of program variables statically
- RIC: Reduced Interval Congruence
  - Interval: [0,4] = {0, 1, 2, 3, 4}
  - RIC: 4[0,4] + 10 = {10, 14, 18, 22, 26}
- Can ensure memory accesses align to memory boundaries

Example


MOV eax, 4
CMP eax, ebx
JE Label
ADD eax, 2
SUB eax, 2
MOV [ADDR], eax
eax = 0[0,0]+4
eax = 0[0,0]+6
eax = 0[0,0]+2
eax = 4[0,1]+2

Operations

- Easily supported operations
  - ADD
  - SUB
  - MOV
  - others
- Unsupported Operations
  - DIV
  - OR
  - AND
  - …

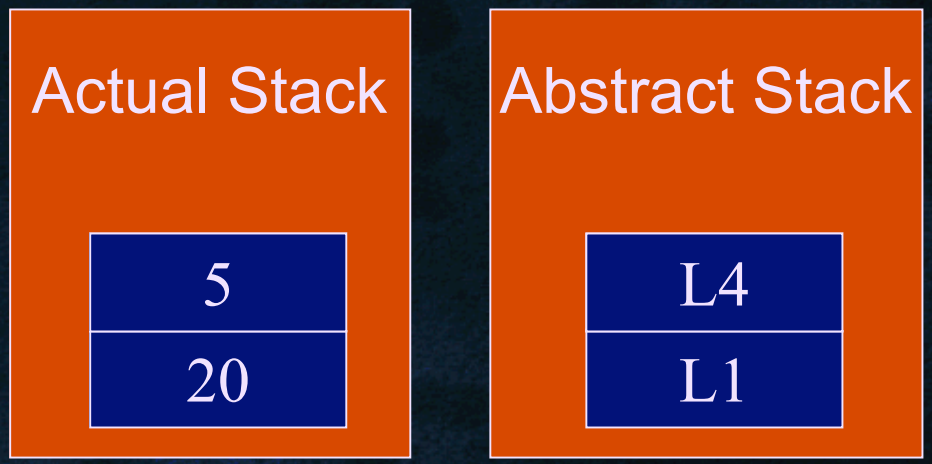
VSA Limitations

- Assumes executable conforms to standard conventions
- Depends on control-flow graph
- We would like to remove these two requirements

Abstract Stack

- Each node stores an instruction address
- Example:

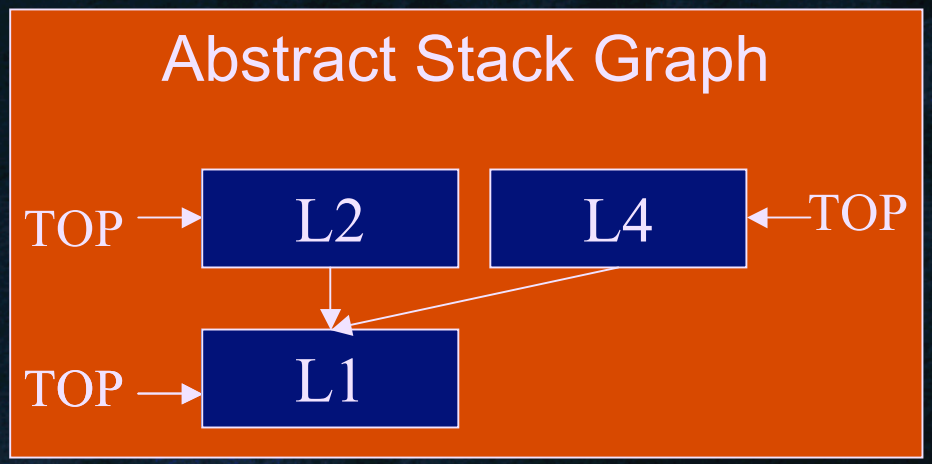
```
L1: PUSH 20
L2: PUSH 10
L3: POP  ebx
L4: PUSH 5
```



Abstract Stack Graph

- Contains all possible abstract stack
- Example:

```
L1: PUSH 20
L2: PUSH 10
L3: POP  ebx
L4: PUSH 5
```



Abstract Stack Graph

- Apply set of rules to detect obfuscations
- Limitations
  - Does not know register/memory contents
  - Can’t handle instructions like:
    - `sub esp, eax`
    - `mov esp, eax`
  - Can’t handle indirect jumps/calls

Talk Contents

- Motivation
- Previous work
  - VSA (Balakrishnan and Reps, 2004)
  - Abstract Stack Graph (Lakhotia and Kumar 2004)
- VSA+ASG
  - Domain
  - Interpretation example
- Application
  - Detect Call obfuscations
- Implementation
- Future and Conclusions

Our Goal

- VSA+ASG
- Achieve:
  - Support more instructions (ex. `sub esp, eax`)
  - Support indirect jumps/calls
  - Not rely on CFG
  - Do not assume binary obeys common standards

Domain

- RIC
- STACK-LOCATION
- VALUE := 2-tuple
  - $RIC_{\top}$
  - $P(\text{STACK-LOCATION})_{\top}$
- STATE := 3-tuple
  - REGISTER $\rightarrow$ VALUE
  - STACK-LOCATION $\rightarrow$ VALUE
  - STACK-LOCATION $\times$ STACK-LOCATION

Operation: Addition

+: VALUE × VALUE × STATE → VALUE

Pairwise addition of both components of VALUE

+: RIC × RIC → RIC

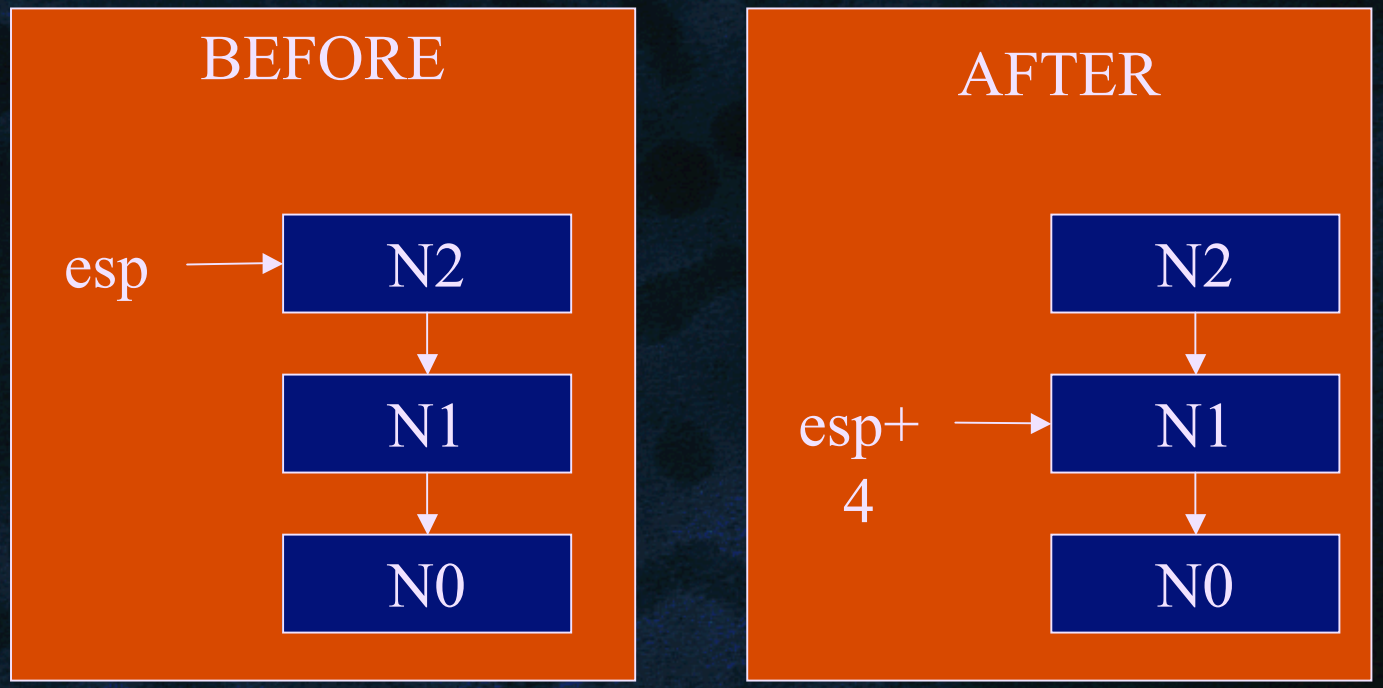
+: STACK-LOCATION × STACK-LOCATION = NULL

+: RIC × STACK-LOCATION → P(STACK-LOCATION)

- RIC+RIC
  - Equals approximation of the sum of each value in the two RICs
- Stack-location + Stack-location
  - Unable to do, since requires knowing actual addresses

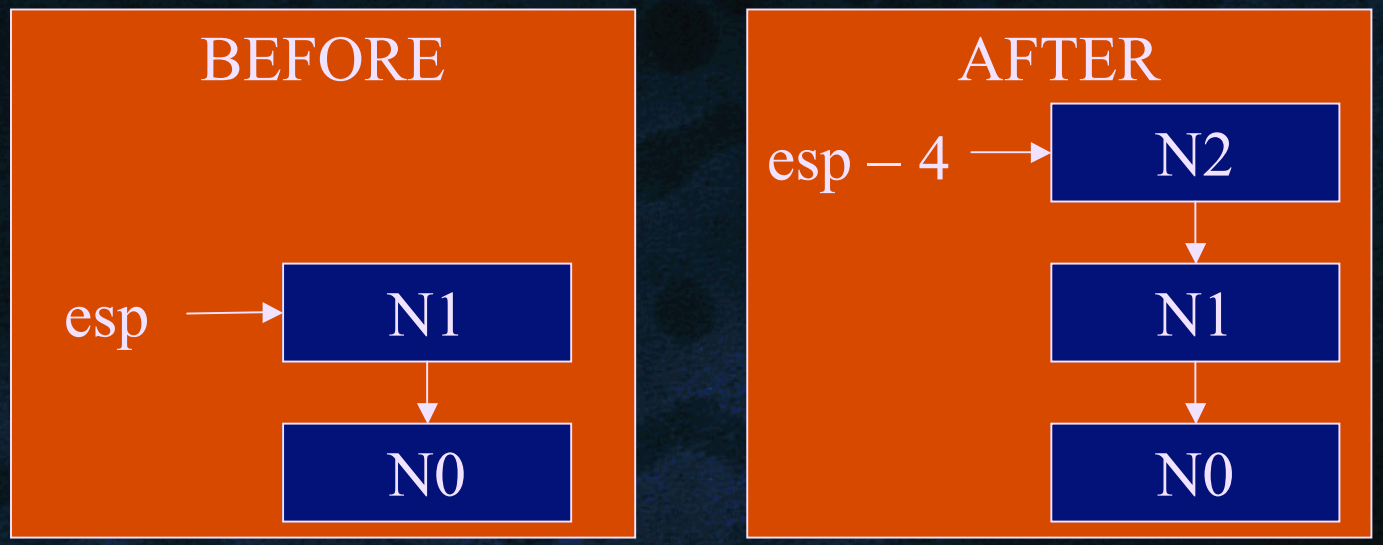
Operation: Addition

- Stack-location + RIC
  - Travel down the stack



Operation: Subtraction

- RIC – RIC, Stack-location – Stack-location
  - Similar to addition
- Stack-location – RIC
  - Travel up the stack or add new nodes



Operation: Memory Operations

- Push
  - Creates a new stack-location
  - Stores pushed value at stack-location
- Pop
  - Retrieves value at top of stack
  - Increments top of stack

Operation: Memory Operations

- Store
  - Places a value in memory
- Load
  - Retrieves a value from memory
- Other operations defined in paper

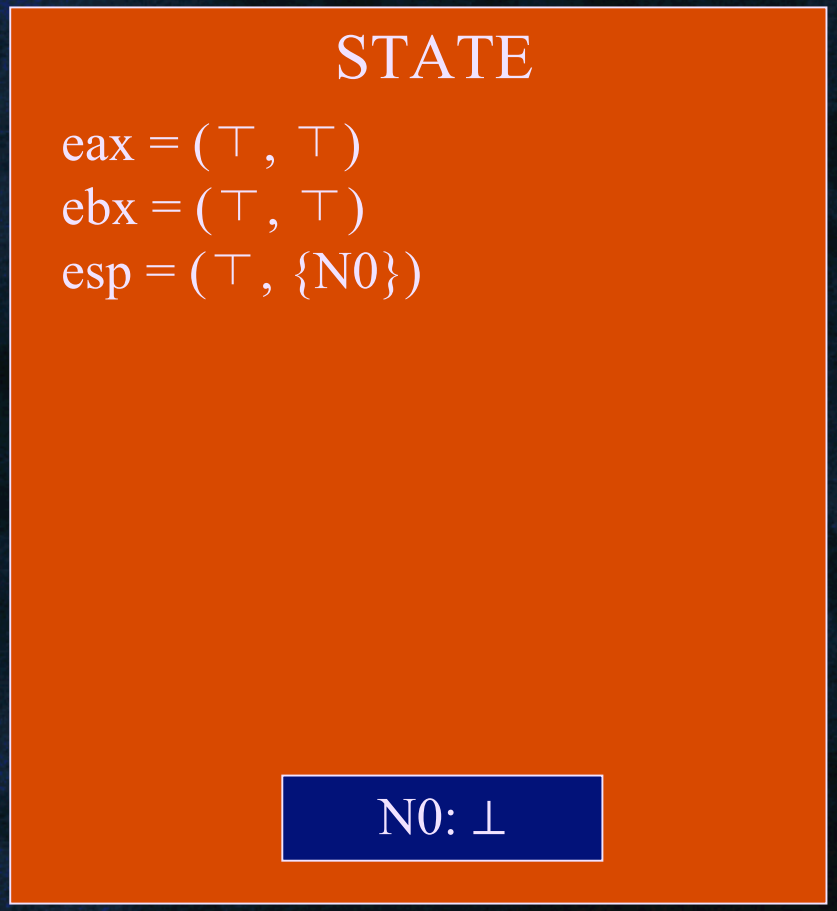
Interpretation Example

- Main
  - Pushes two values onto stack
  - Calls Max
- Max
  - Places max of parameters in eax

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```

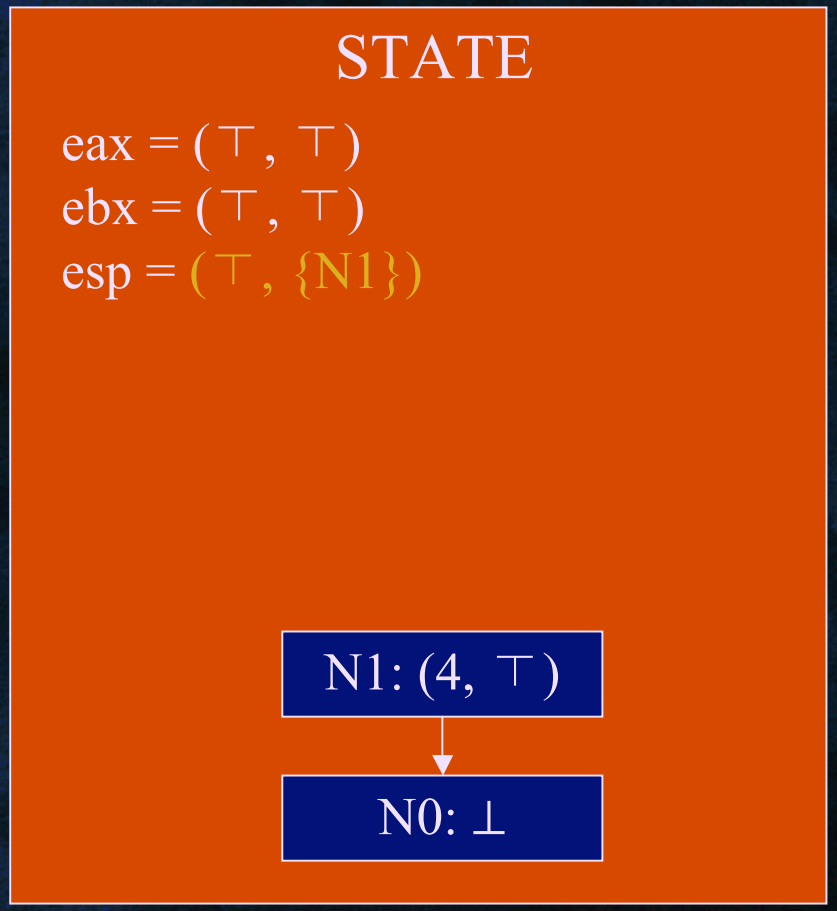
Interpretation Example

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```



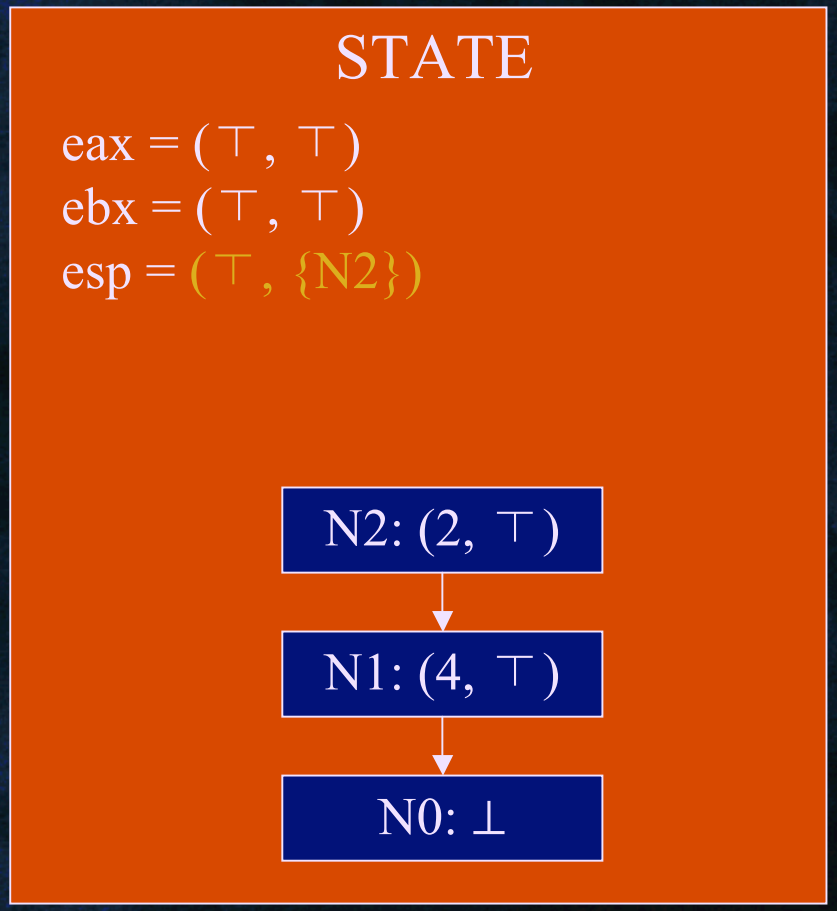
Interpretation Example

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```



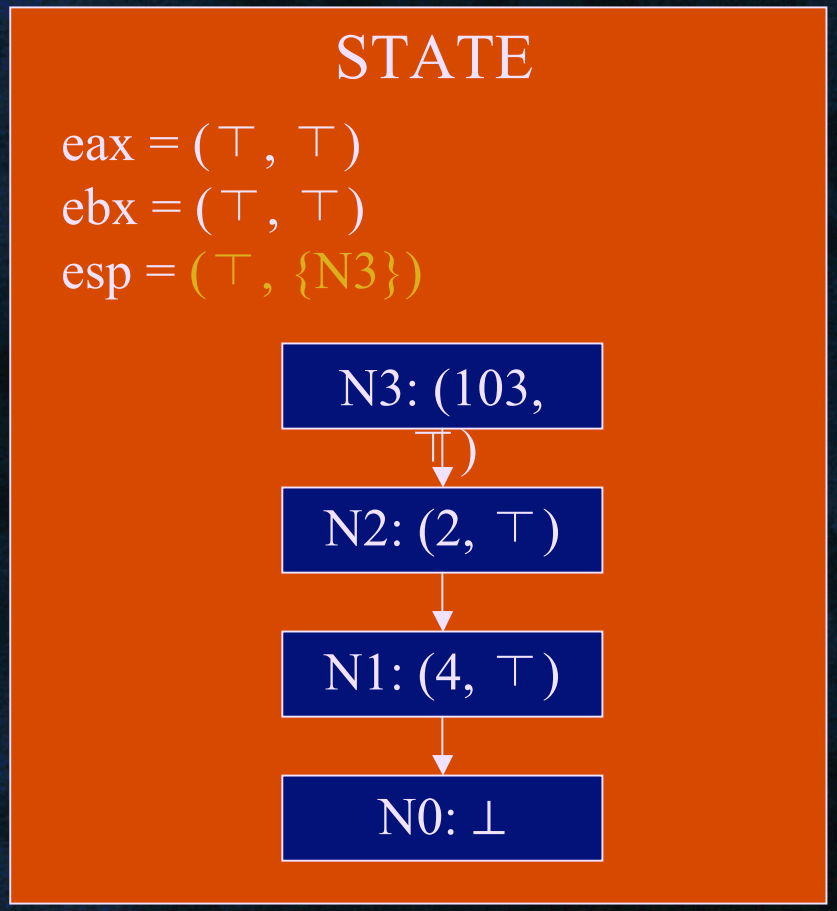
Interpretation Example

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```



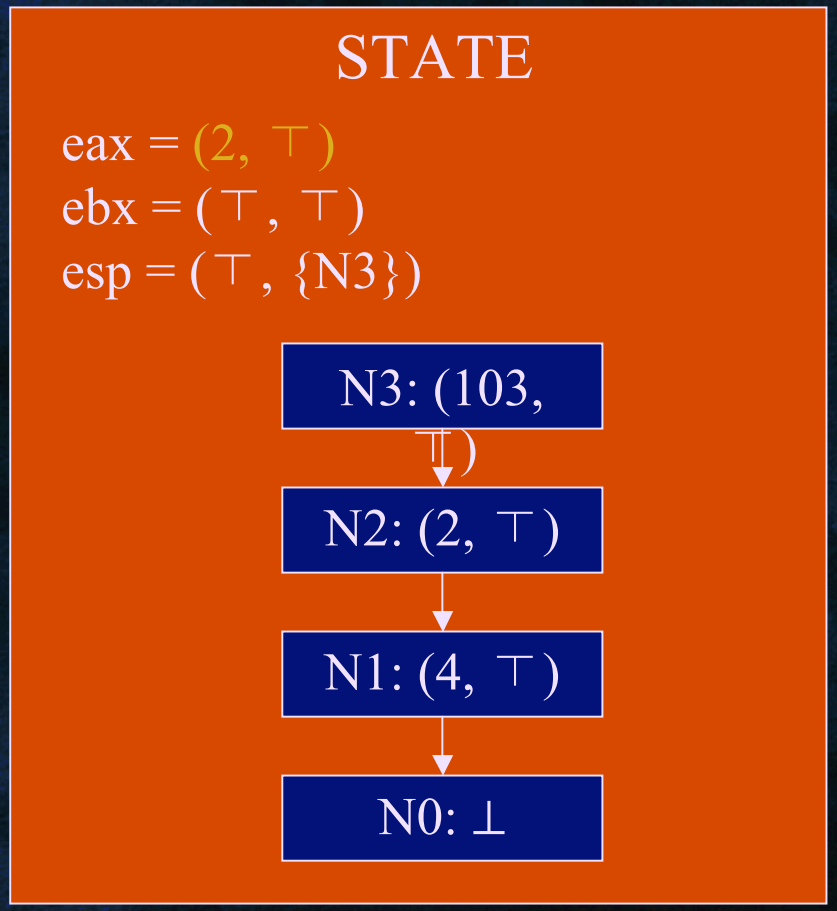
Interpretation Example

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```



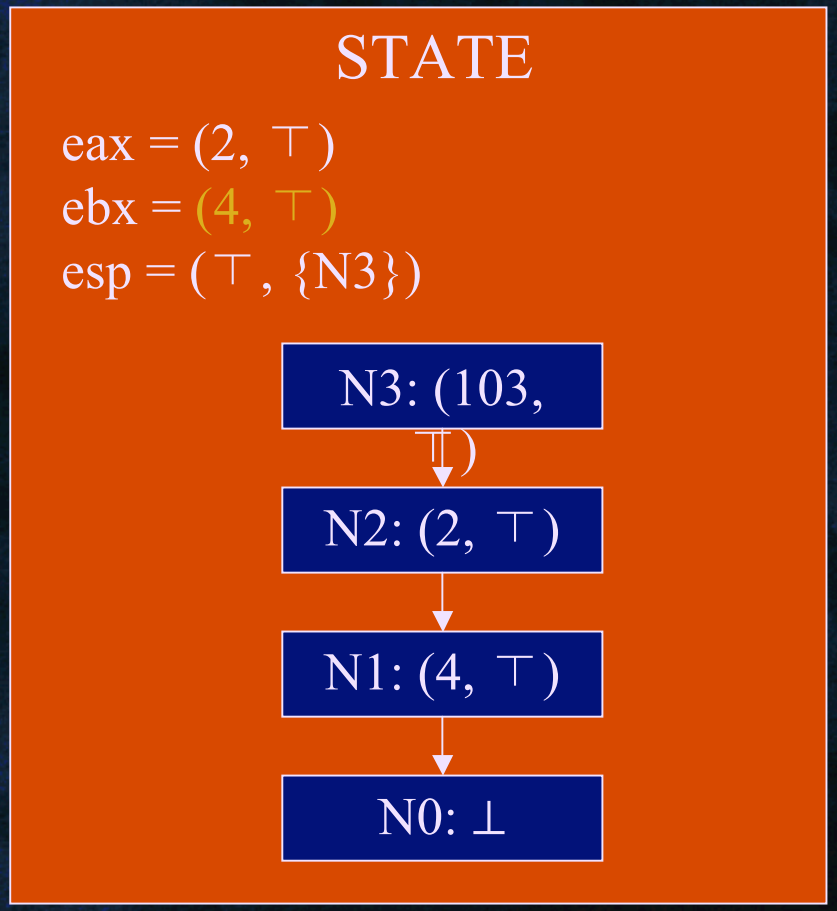
Interpretation Example

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```



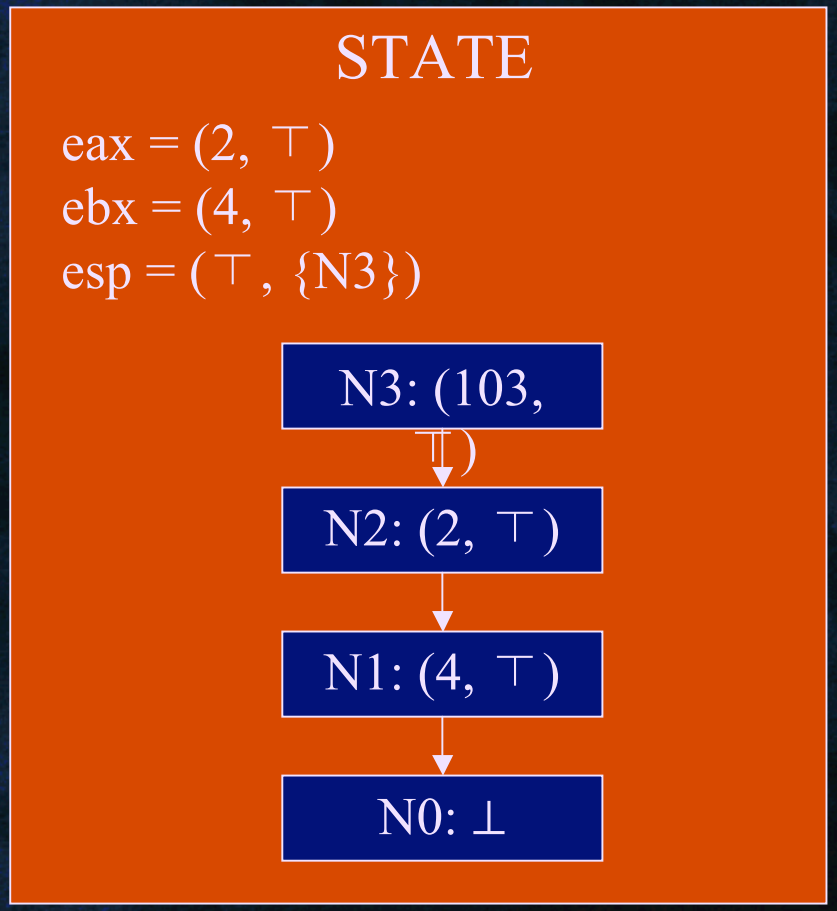
Interpretation Example

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```



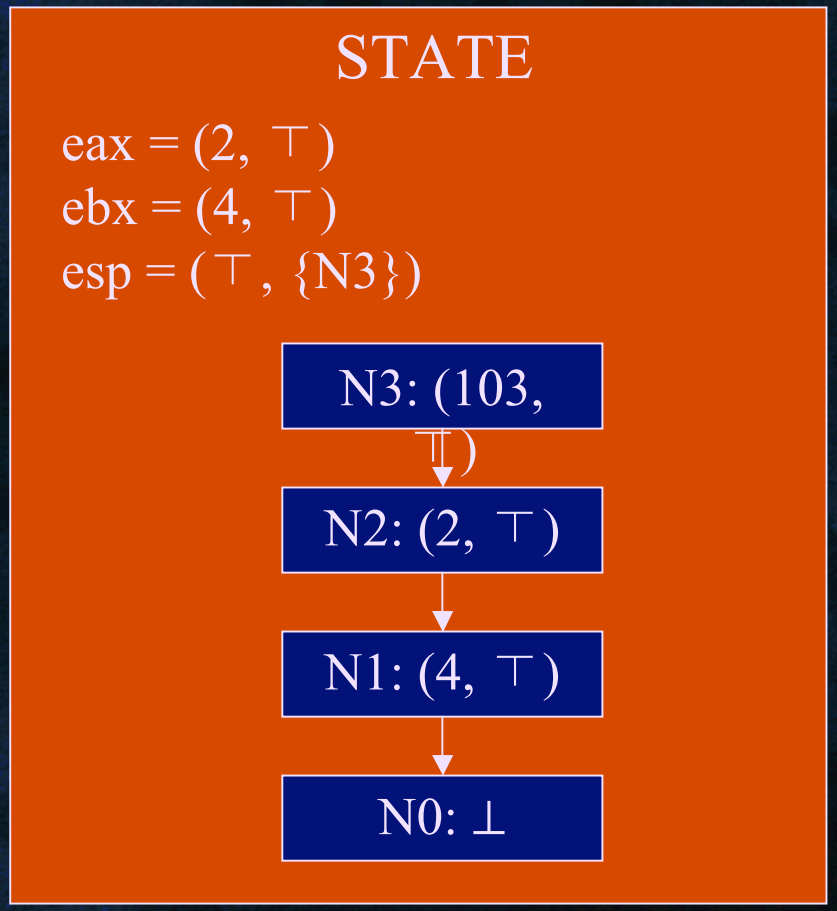
Interpretation Example

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```



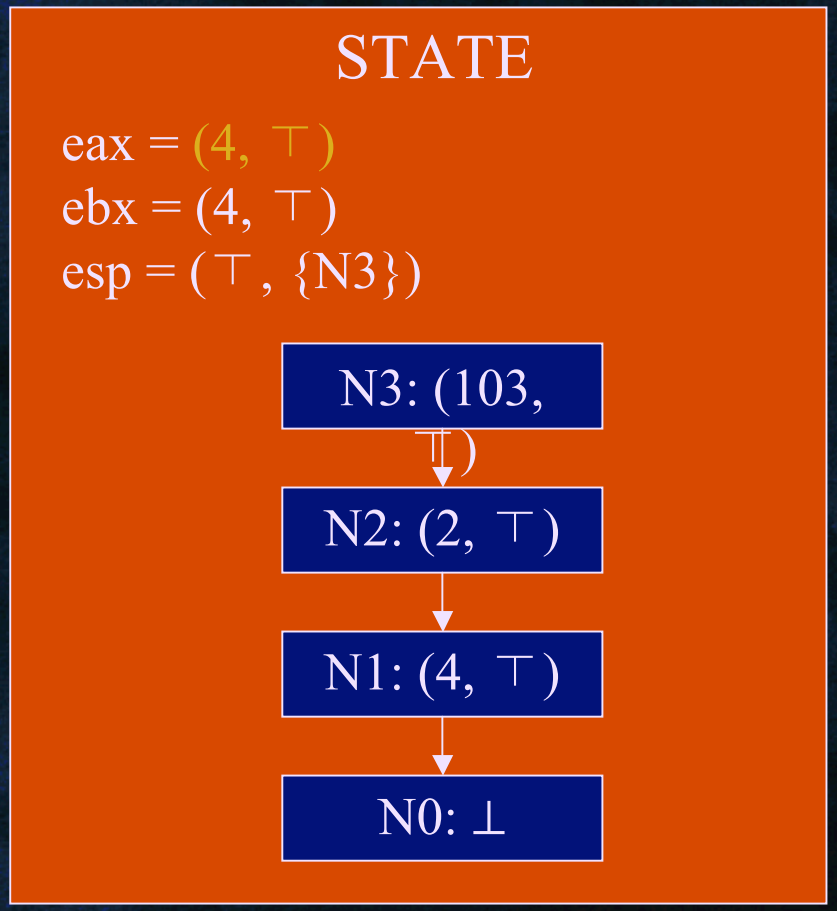
Interpretation Example

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```



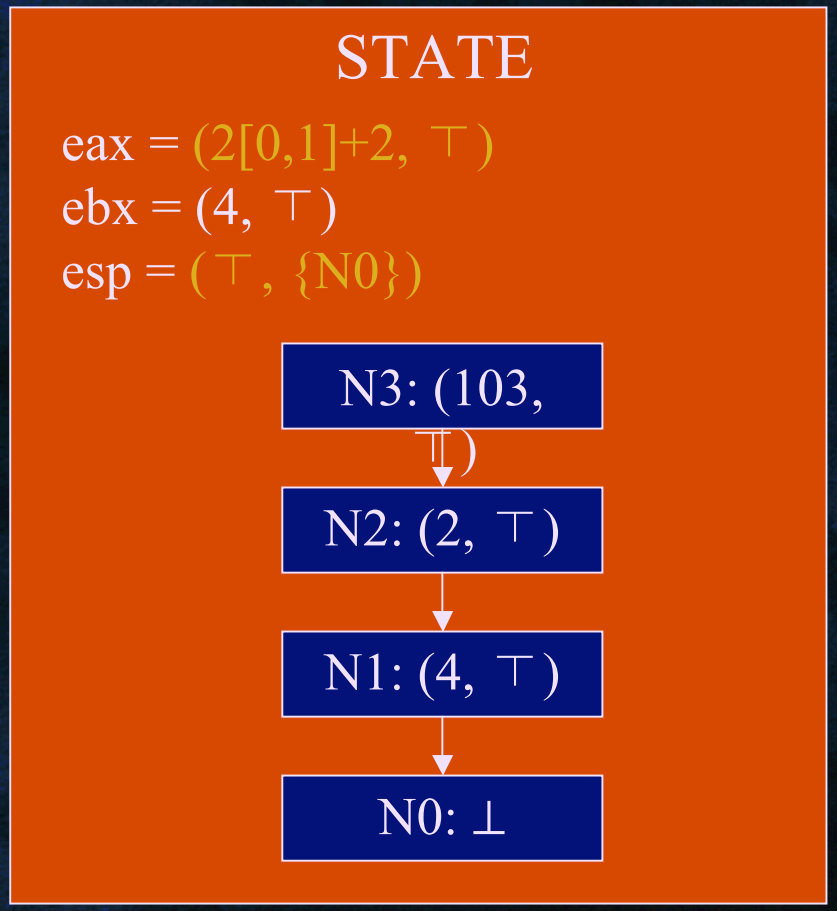
Interpretation Example

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```



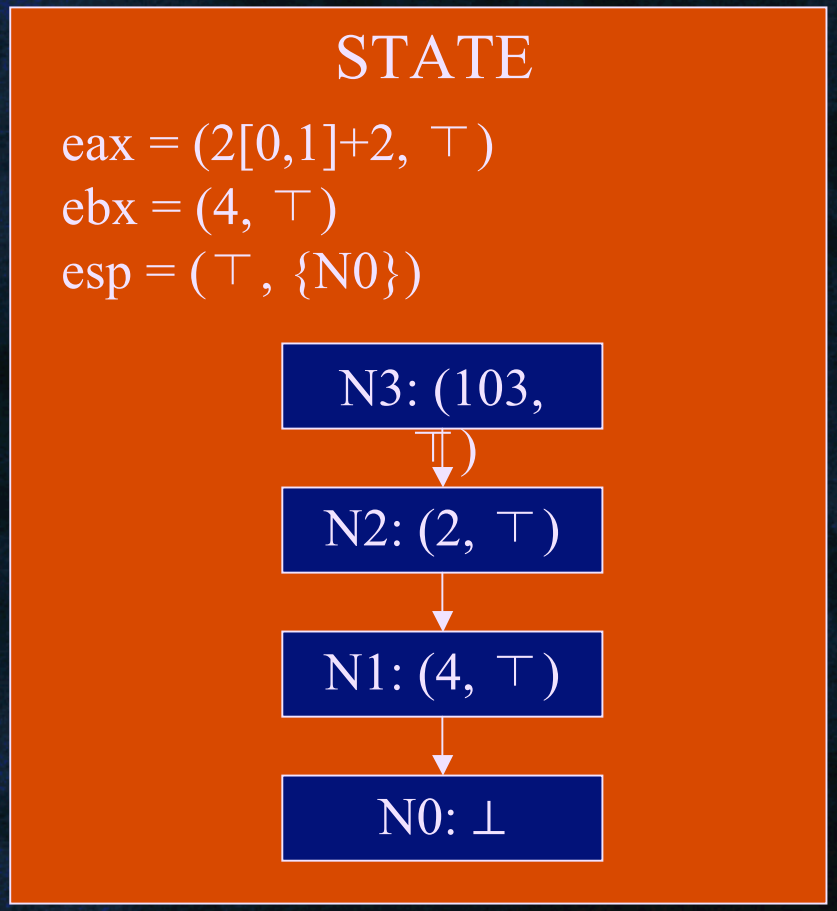
Interpretation Example

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```



Interpretation Example

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    CALL   Max
103    …

Max:
104    MOV    eax, [esp+4]
105    MOV    ebx, [esp+8]
106    CMP    eax, ebx
107    JG     Done
108    MOV    eax, ebx

Done:
109    RET    8
```



Talk Contents

- Motivation
- Previous work
  - VSA (Balakrishnan and Reps, 2004)
  - Abstract Stack Graph (Lakhotia and Kumar 2004)
- VSA+ASG
  - Domain
  - Interpretation example
- Application
  - Detect Call obfuscations
- Implementation
- Future and Conclusions

Application

- Detect
  - Call replaced with equivalent instruction(s)
  - Return address manually removed from stack
- Flag as obfuscation if
  - instruction = `retn` AND topOfStack.creator ≠ `call`
  - instruction = `pop` AND topOfStack.creator = `call`
- Rules are applied to each instruction

Replace CALL with JMP

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```

STATE

eax = (⊤, ⊤)
ebx = (⊤, ⊤)
esp = (⊤, {N0})

N0: ⊥

Replace CALL with JMP

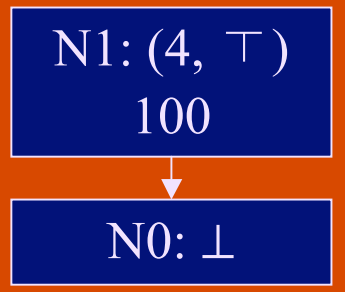
```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```

STATE

eax = (⊤, ⊤)

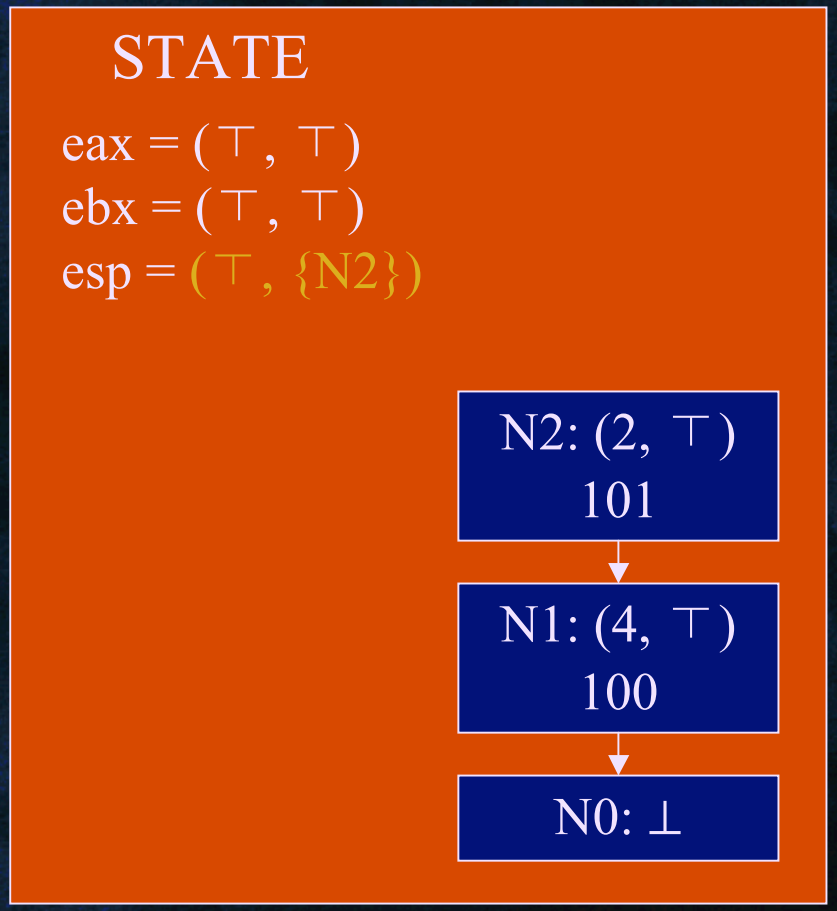
ebx = (⊤, ⊤)

esp = (⊤, {N1})



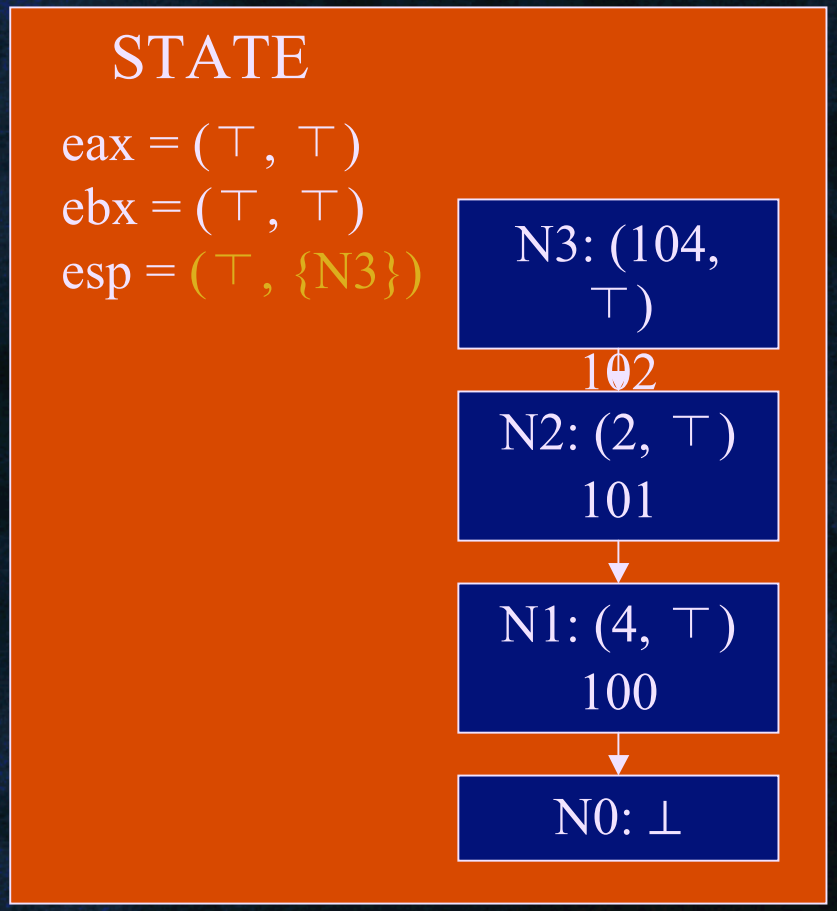
Replace CALL with JMP

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```



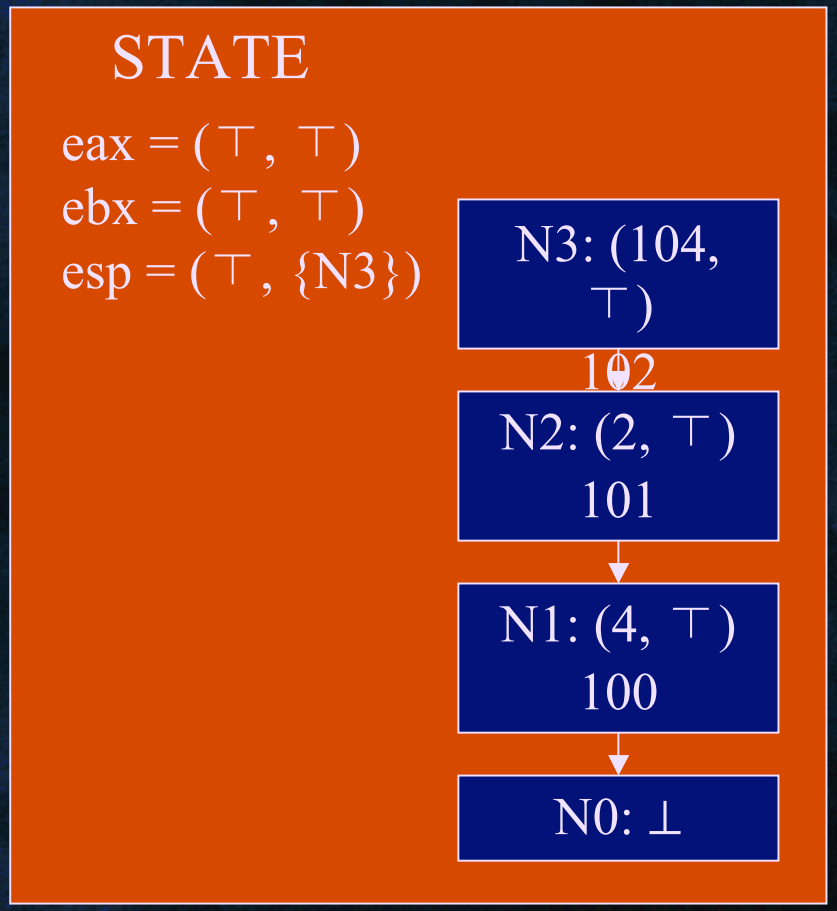
Replace CALL with JMP

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```



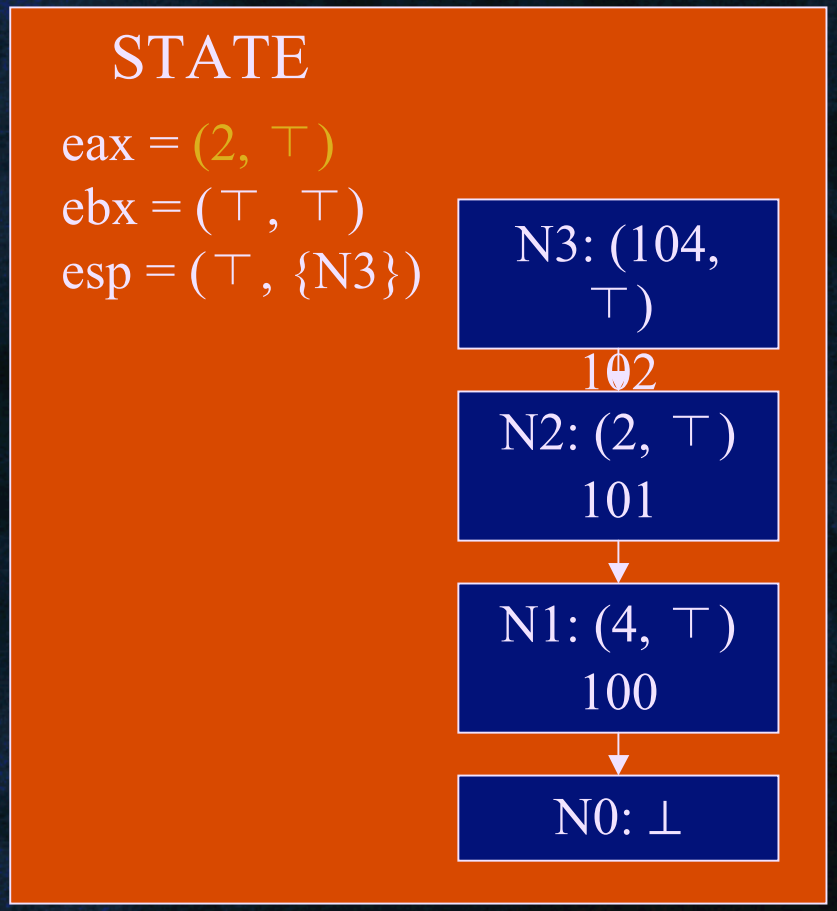
Replace CALL with JMP

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```



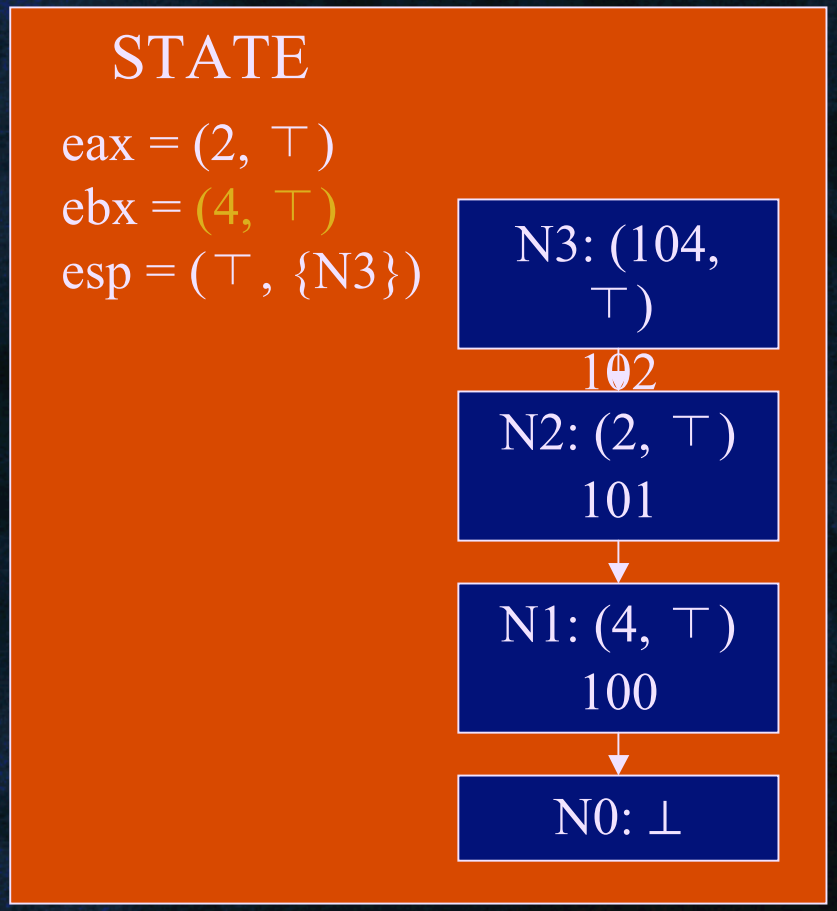
Replace CALL with JMP

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```



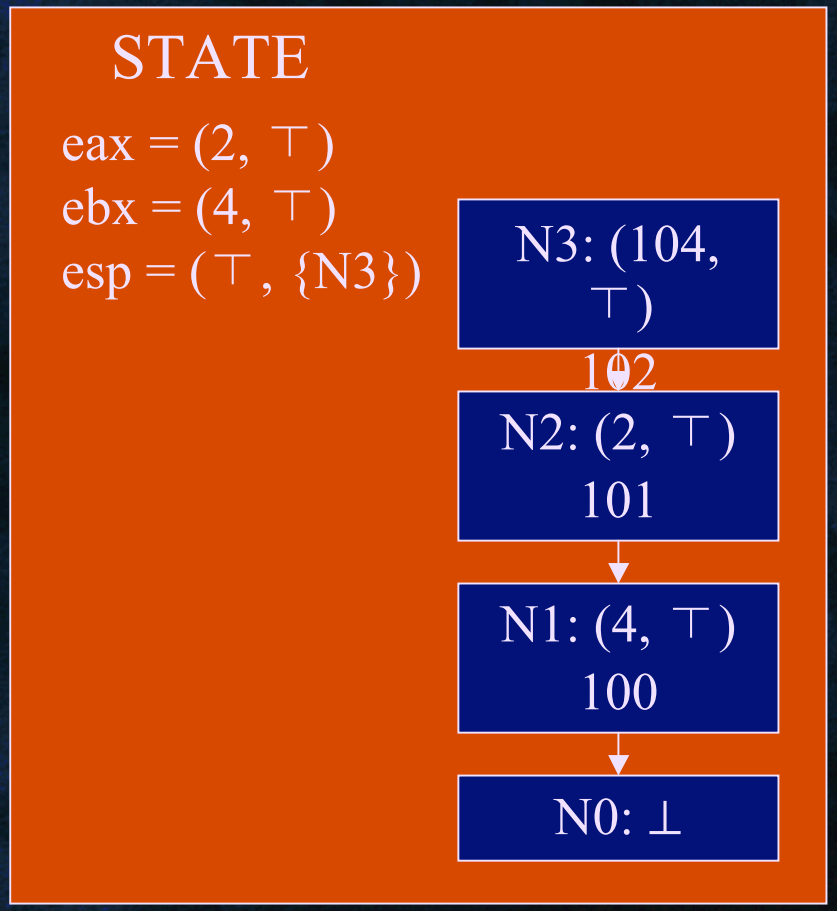
Replace CALL with JMP

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```



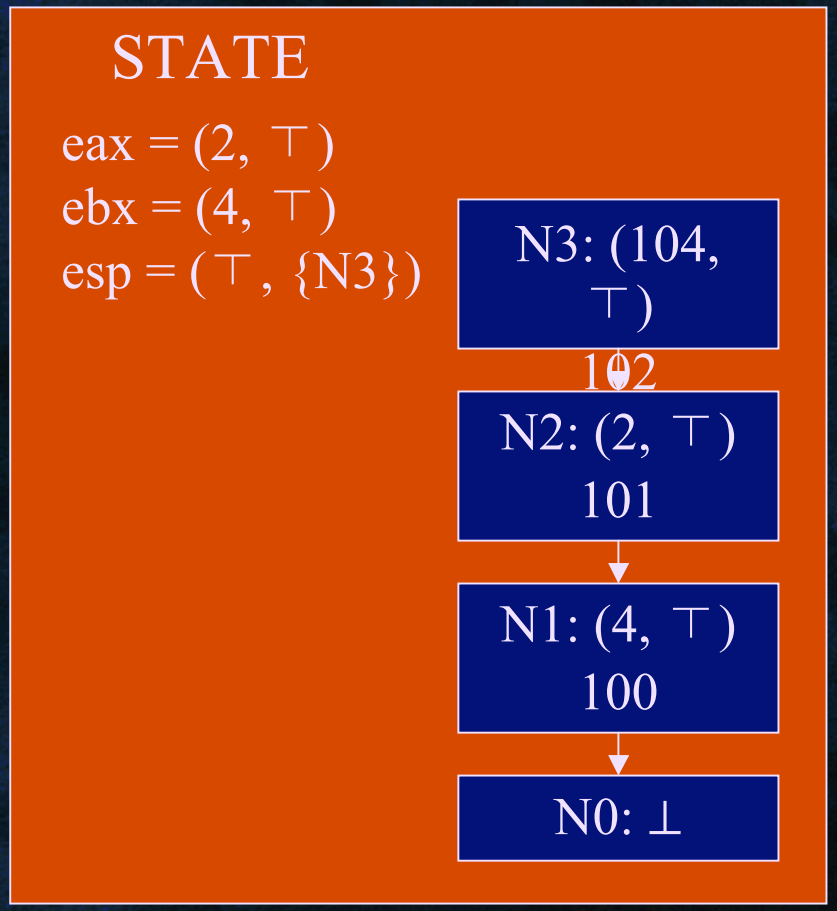
Replace CALL with JMP

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```



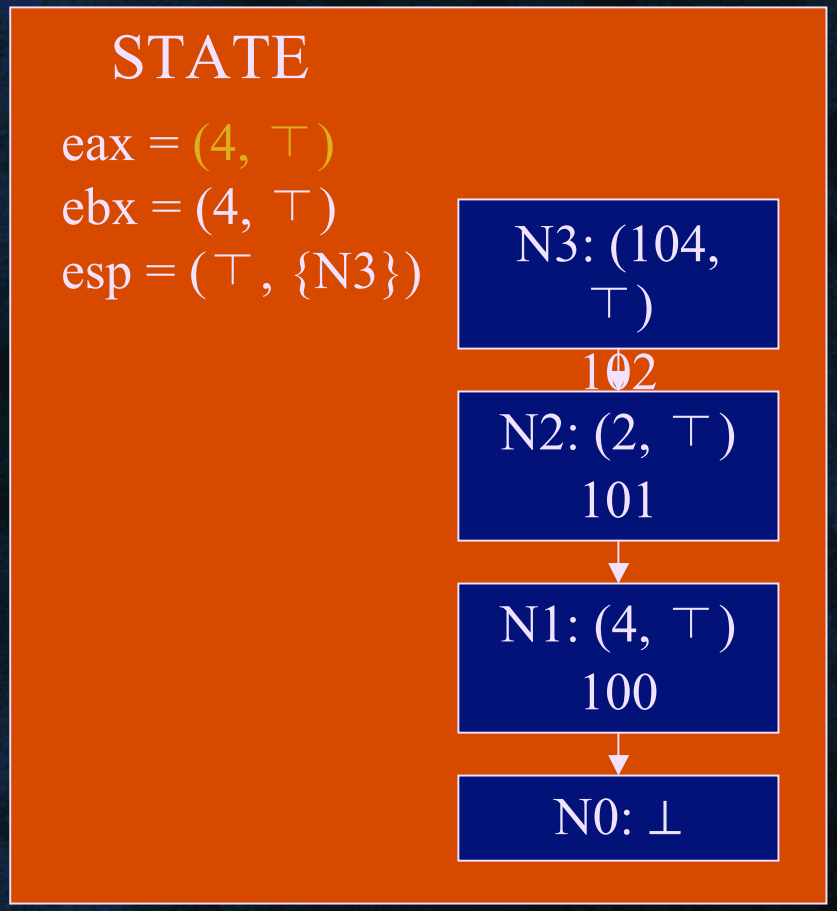
Replace CALL with JMP

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```



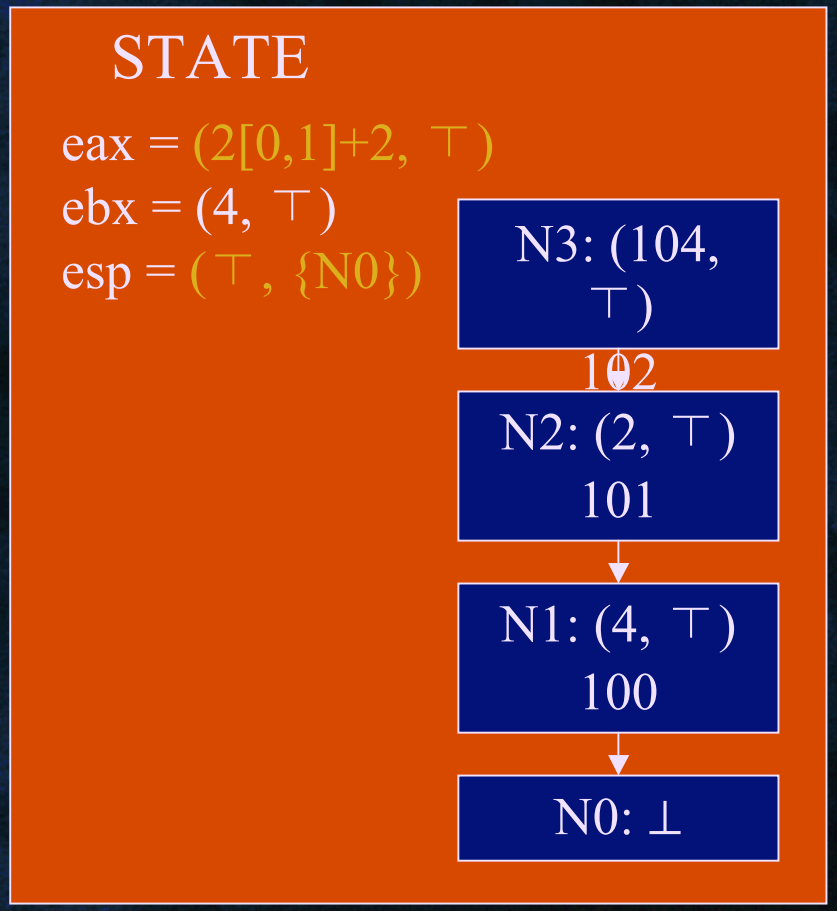
Replace CALL with JMP

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```



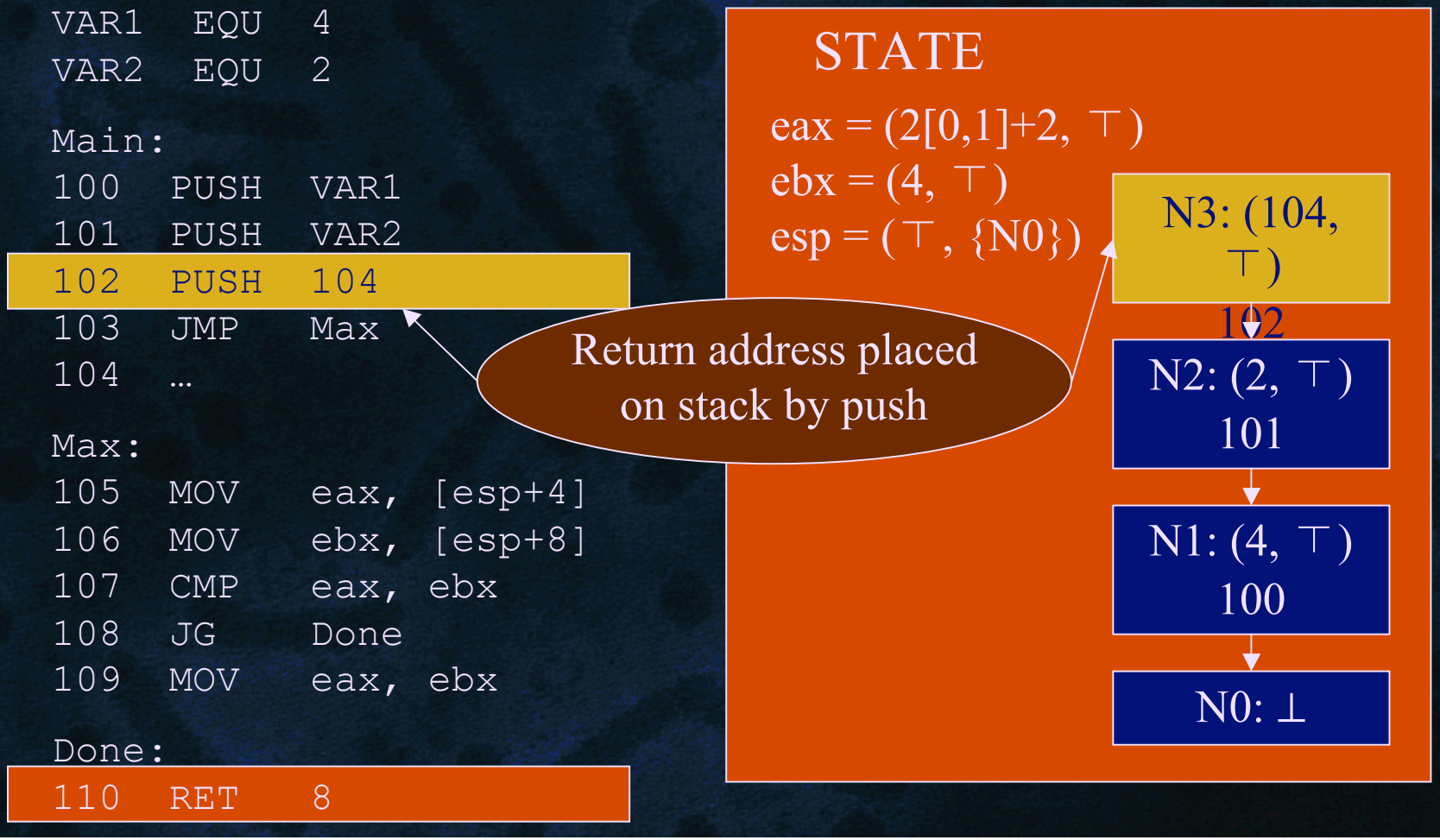
Replace CALL with JMP

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```



Replace CALL with JMP

```
VAR1   EQU   4
VAR2   EQU   2

Main:
100    PUSH   VAR1
101    PUSH   VAR2
102    PUSH   104
103    JMP    Max
104    …

Max:
105    MOV    eax, [esp+4]
106    MOV    ebx, [esp+8]
107    CMP    eax, ebx
108    JG     Done
109    MOV    eax, ebx

Done:
110    RET    8
```



Removal of Return Address

```
Main:
100   PUSH   4
101   PUSH   2
102   CALL   Max
103   …

Max:
104   MOV    eax, [esp+4]
105   MOV    ebx, [esp+8]
106   CMP    eax, ebx
107   JG     Done
108   MOV    eax, ebx

Done:
109   POP    ebx
110   ADD    esp, 8
111   JMP    ebx
```

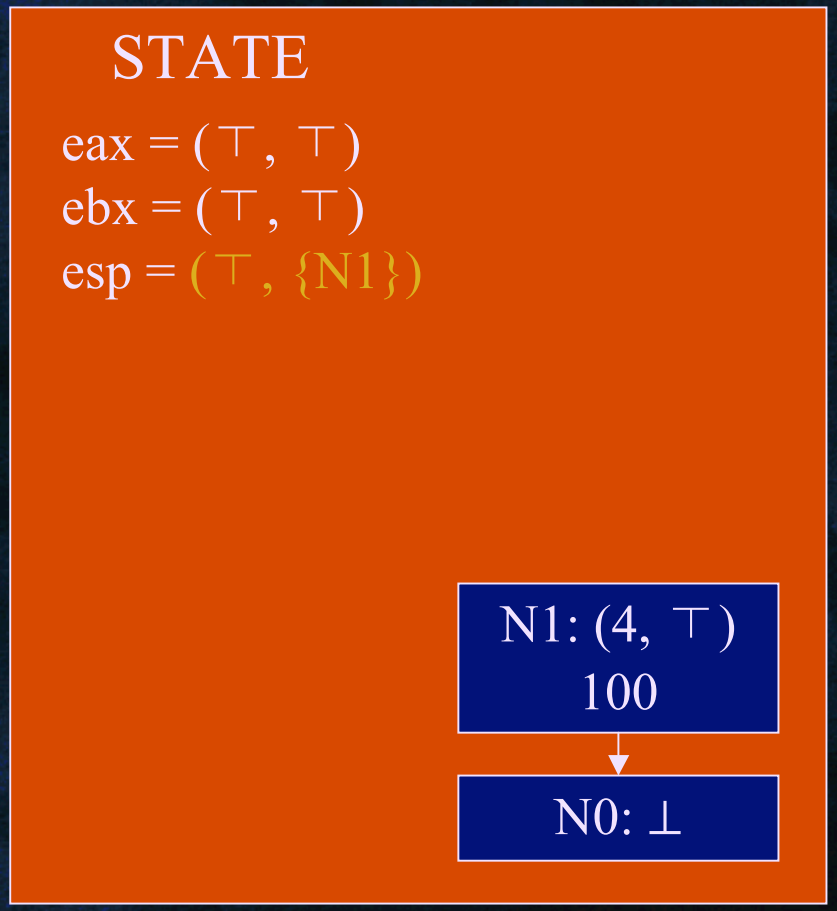
STATE

eax = (⊤, ⊤)
ebx = (⊤, ⊤)
esp = (⊤, {N0})

N0: ⊥

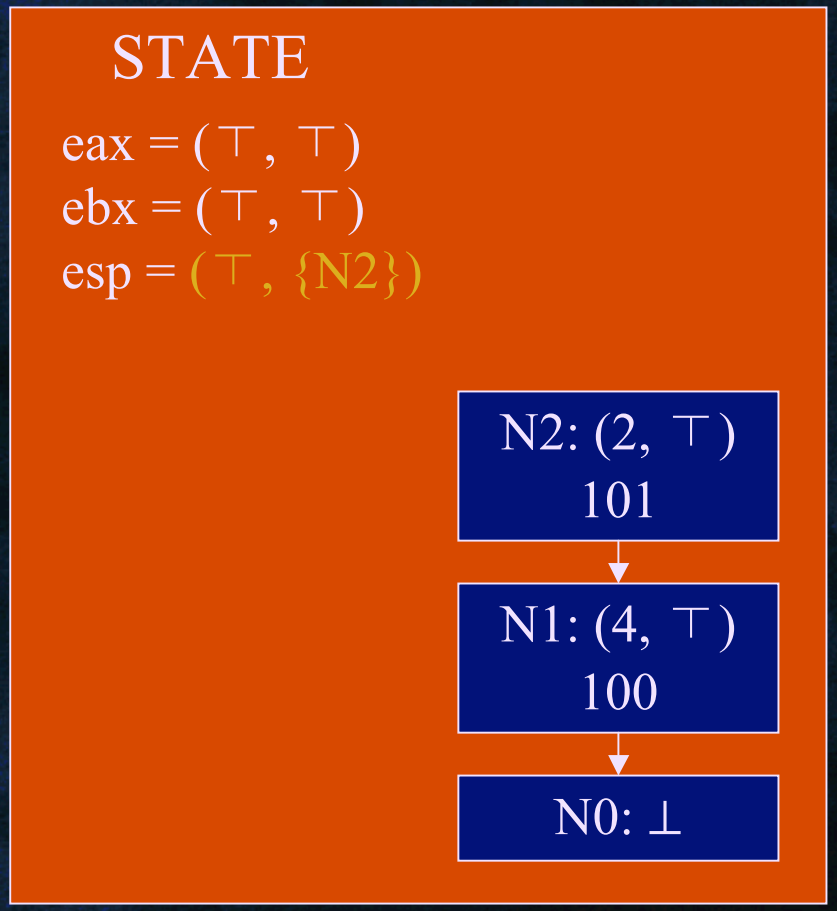
Removal of Return Address

```
Main:
100   PUSH   4
101   PUSH   2
102   CALL   Max
103   …

Max:
104   MOV    eax, [esp+4]
105   MOV    ebx, [esp+8]
106   CMP    eax, ebx
107   JG     Done
108   MOV    eax, ebx

Done:
109   POP    ebx
110   ADD    esp, 8
111   JMP    ebx
```



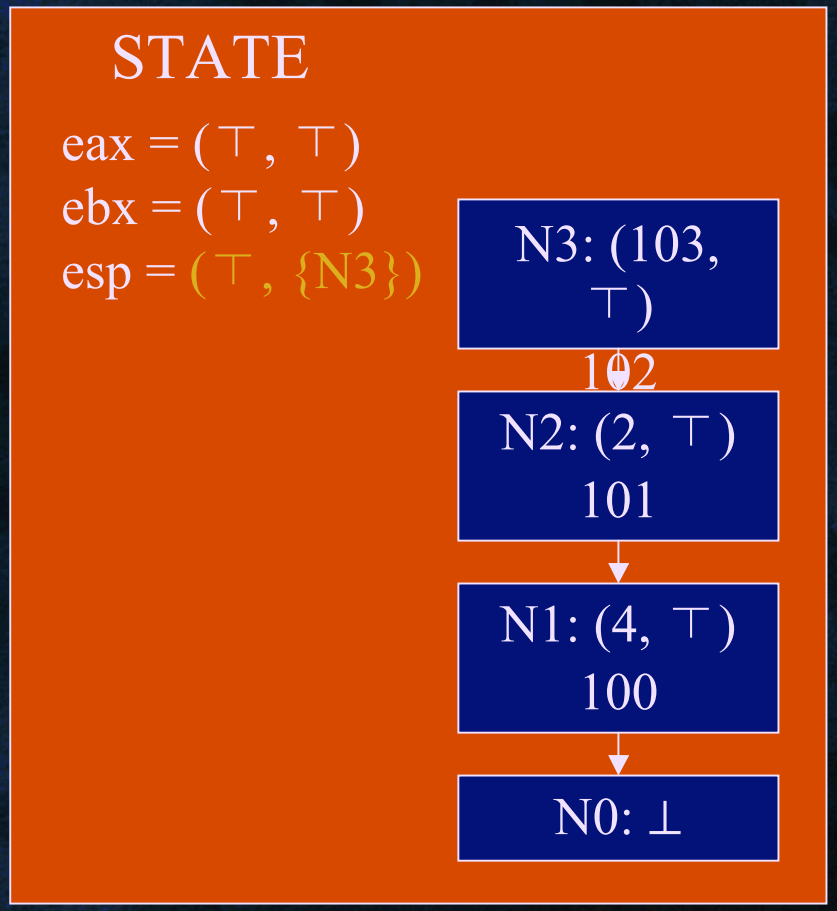
Removal of Return Address

```
Main:
100   PUSH   4
101   PUSH   2
102   CALL   Max
103   …

Max:
104   MOV    eax, [esp+4]
105   MOV    ebx, [esp+8]
106   CMP    eax, ebx
107   JG     Done
108   MOV    eax, ebx

Done:
109   POP    ebx
110   ADD    esp, 8
111   JMP    ebx
```



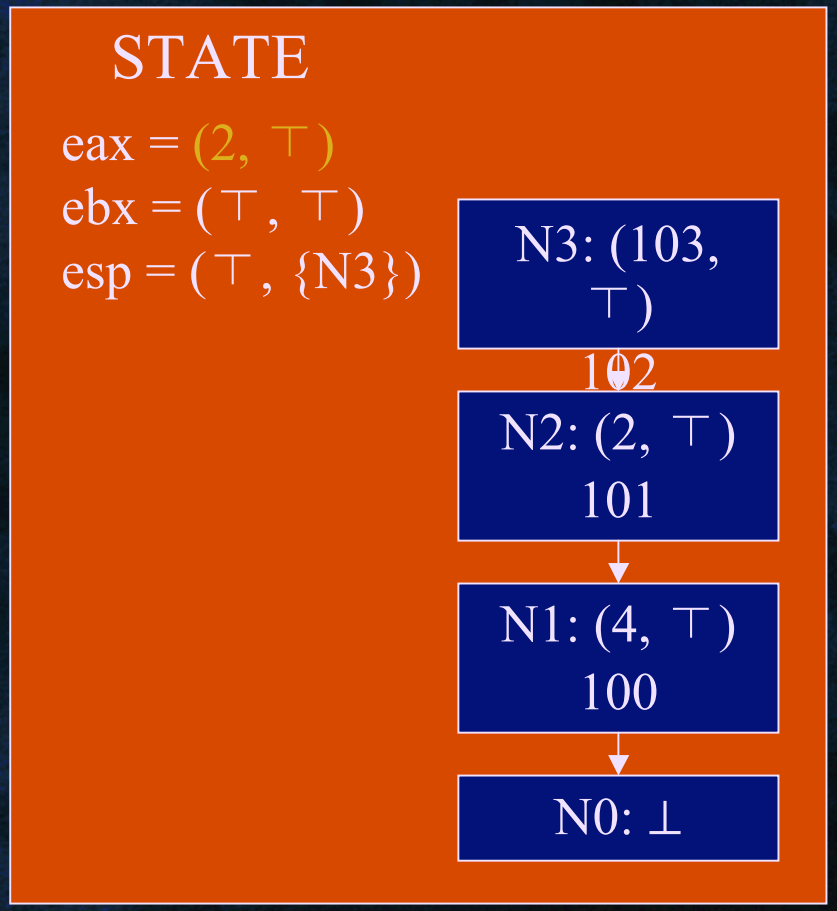
Removal of Return Address

```
Main:
100   PUSH   4
101   PUSH   2
102   CALL   Max
103   …

Max:
104   MOV    eax, [esp+4]
105   MOV    ebx, [esp+8]
106   CMP    eax, ebx
107   JG     Done
108   MOV    eax, ebx

Done:
109   POP    ebx
110   ADD    esp, 8
111   JMP    ebx
```



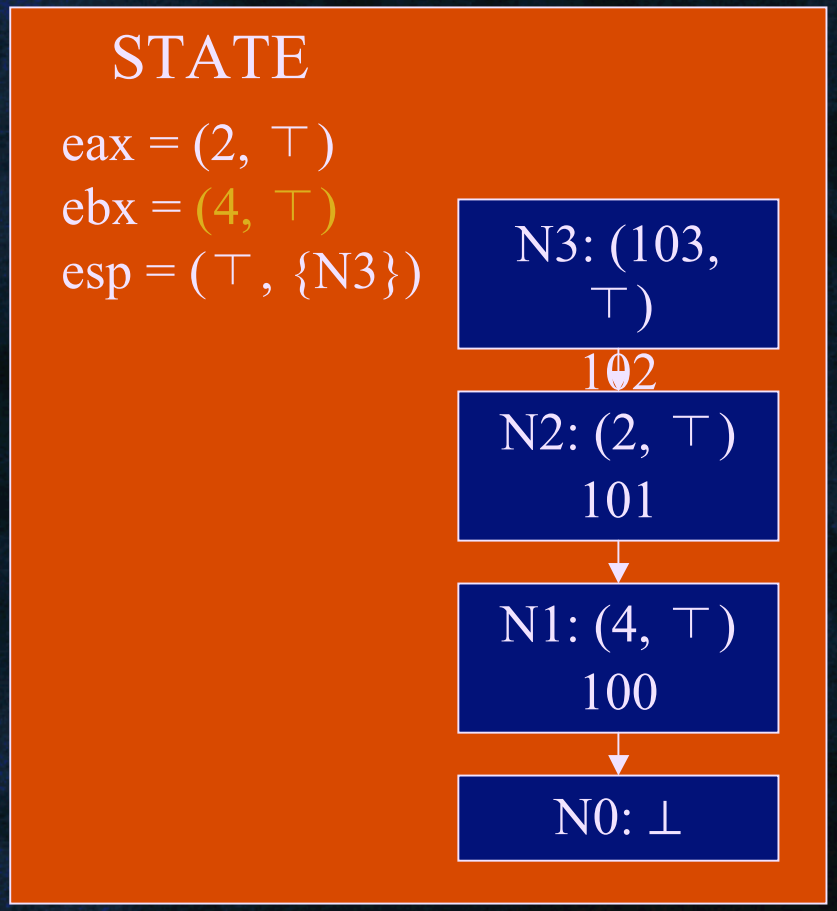
Removal of Return Address

```
Main:
100   PUSH   4
101   PUSH   2
102   CALL   Max
103   …

Max:
104   MOV    eax, [esp+4]
105   MOV    ebx, [esp+8]
106   CMP    eax, ebx
107   JG     Done
108   MOV    eax, ebx

Done:
109   POP    ebx
110   ADD    esp, 8
111   JMP    ebx
```



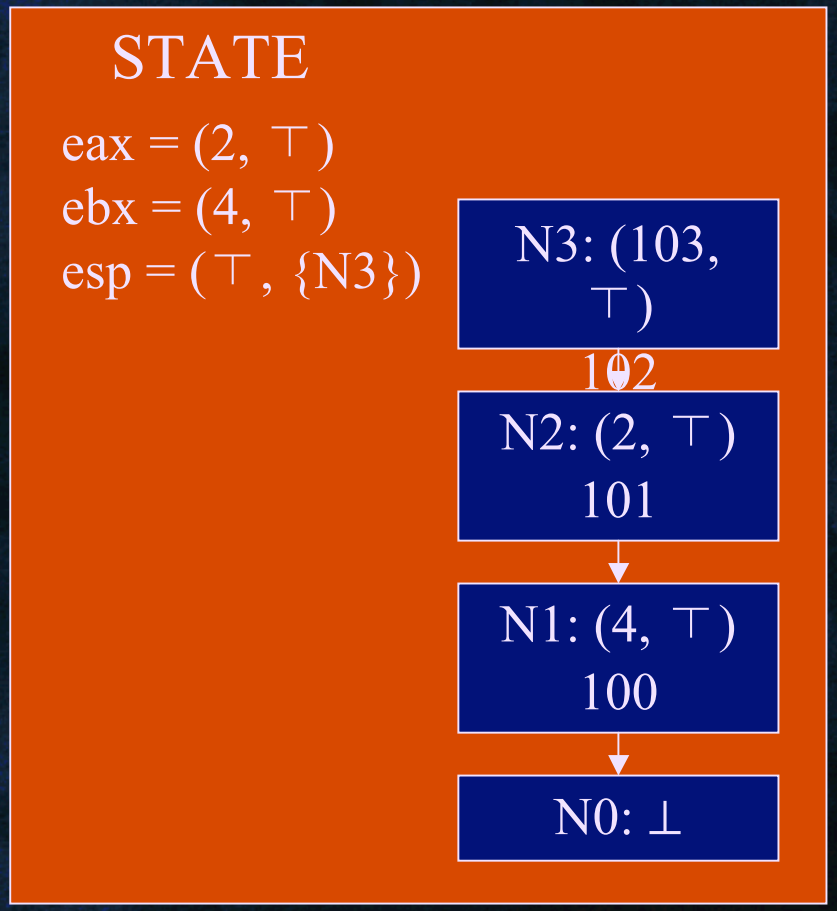
Removal of Return Address

```
Main:
100   PUSH   4
101   PUSH   2
102   CALL   Max
103   …

Max:
104   MOV    eax, [esp+4]
105   MOV    ebx, [esp+8]
106   CMP    eax, ebx
107   JG     Done
108   MOV    eax, ebx

Done:
109   POP    ebx
110   ADD    esp, 8
111   JMP    ebx
```



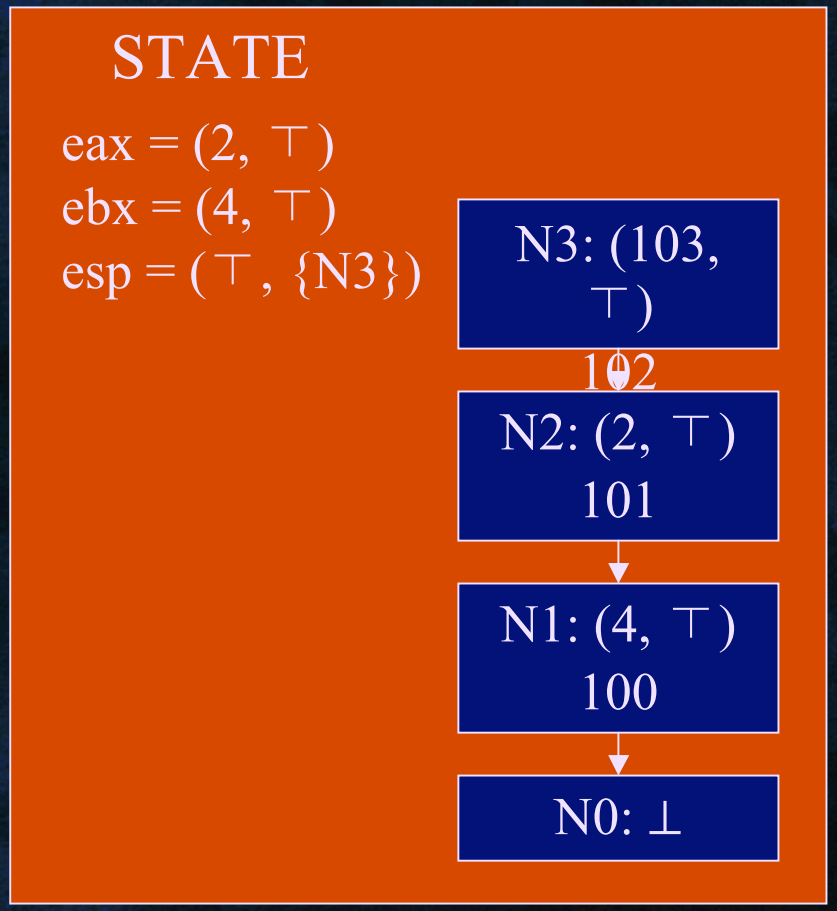
Removal of Return Address

```
Main:
100   PUSH   4
101   PUSH   2
102   CALL   Max
103   …

Max:
104   MOV    eax, [esp+4]
105   MOV    ebx, [esp+8]
106   CMP    eax, ebx
107   JG     Done
108   MOV    eax, ebx

Done:
109   POP    ebx
110   ADD    esp, 8
111   JMP    ebx
```



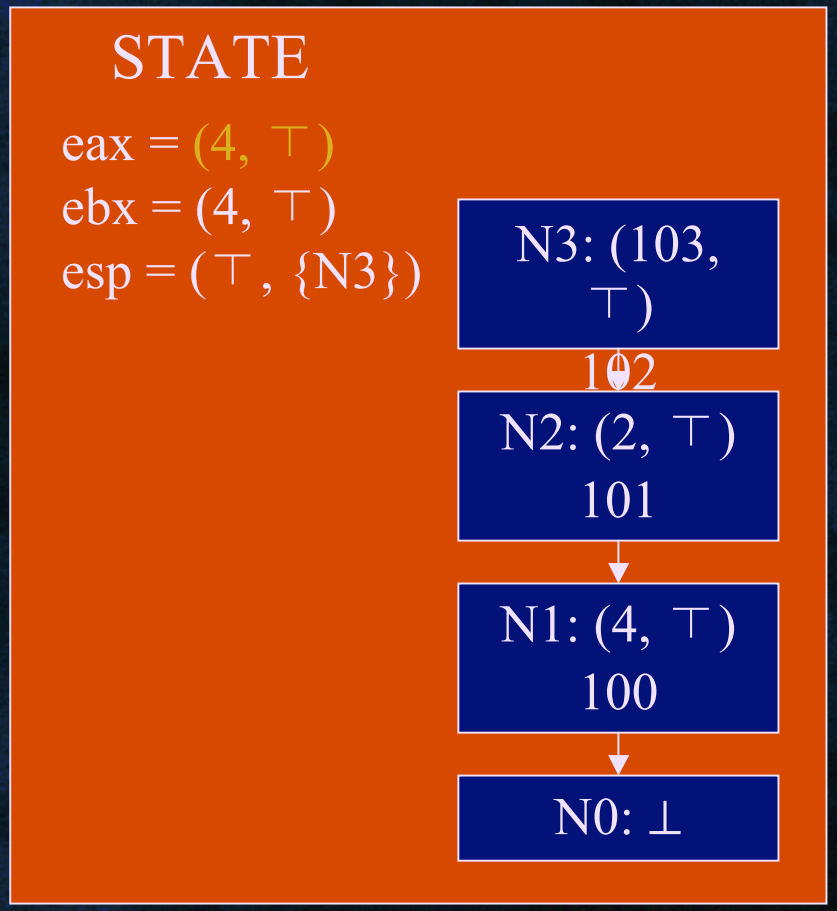
Removal of Return Address

```
Main:
100   PUSH   4
101   PUSH   2
102   CALL   Max
103   …

Max:
104   MOV    eax, [esp+4]
105   MOV    ebx, [esp+8]
106   CMP    eax, ebx
107   JG     Done
108   MOV    eax, ebx

Done:
109   POP    ebx
110   ADD    esp, 8
111   JMP    ebx
```



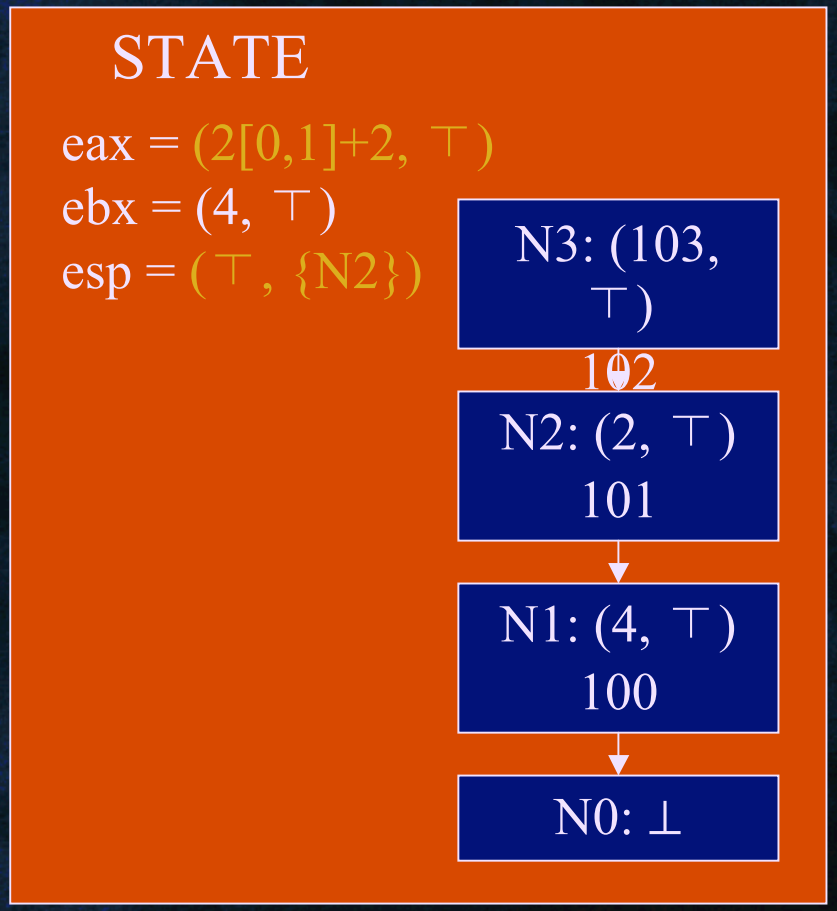
Removal of Return Address

```
Main:
100   PUSH   4
101   PUSH   2
102   CALL   Max
103   …

Max:
104   MOV    eax, [esp+4]
105   MOV    ebx, [esp+8]
106   CMP    eax, ebx
107   JG     Done
108   MOV    eax, ebx

Done:
109   POP    ebx
110   ADD    esp, 8
111   JMP    ebx
```



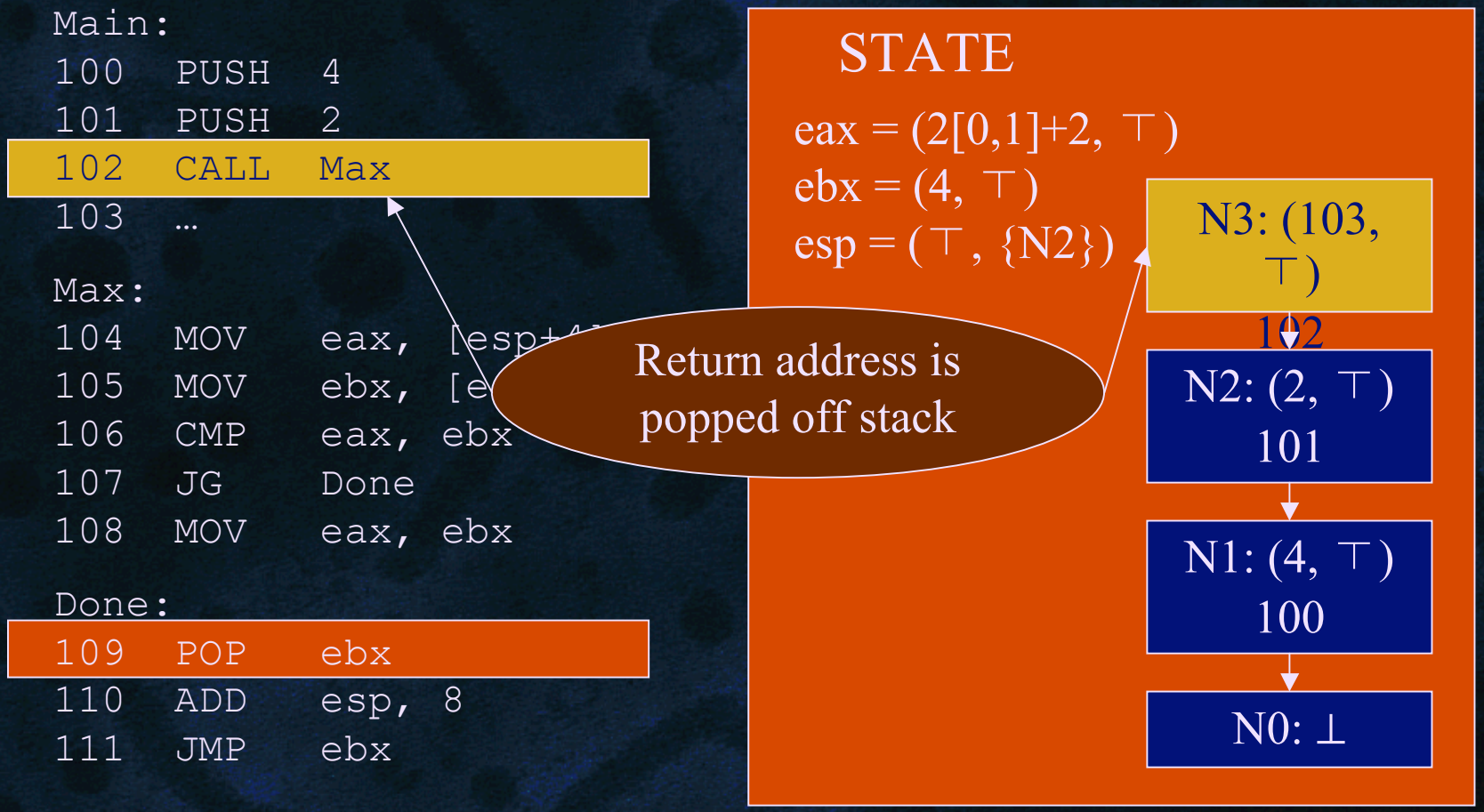
Removal of Return Address

```
Main:
100   PUSH   4
101   PUSH   2
102   CALL   Max
103   …

Max:
104   MOV    eax, [esp+4]
105   MOV    ebx, [esp+8]
106   CMP    eax, ebx
107   JG     Done
108   MOV    eax, ebx

Done:
109   POP    ebx
110   ADD    esp, 8
111   JMP    ebx
```



Removal of Return Address

```
Main:
100   PUSH   4
101   PUSH   2
102   CALL   Max
103   …

Max:
104   MOV    eax, [esp+4
105   MOV    ebx, [e
106   CMP    eax, ebx
107   JG     Done
108   MOV    eax, ebx

Done:
109   POP    ebx
110   ADD    esp, 8
111   JMP    ebx
```

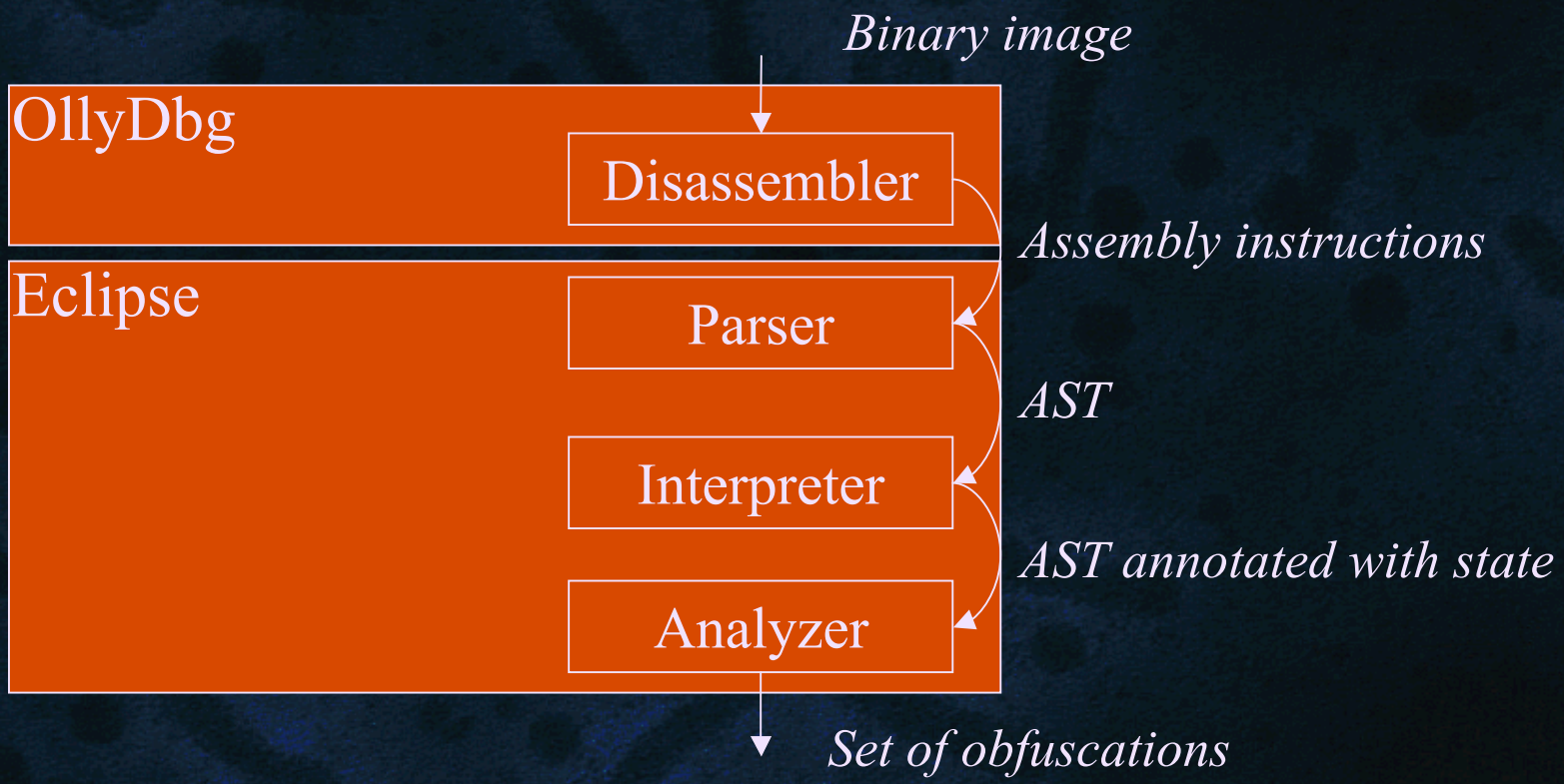


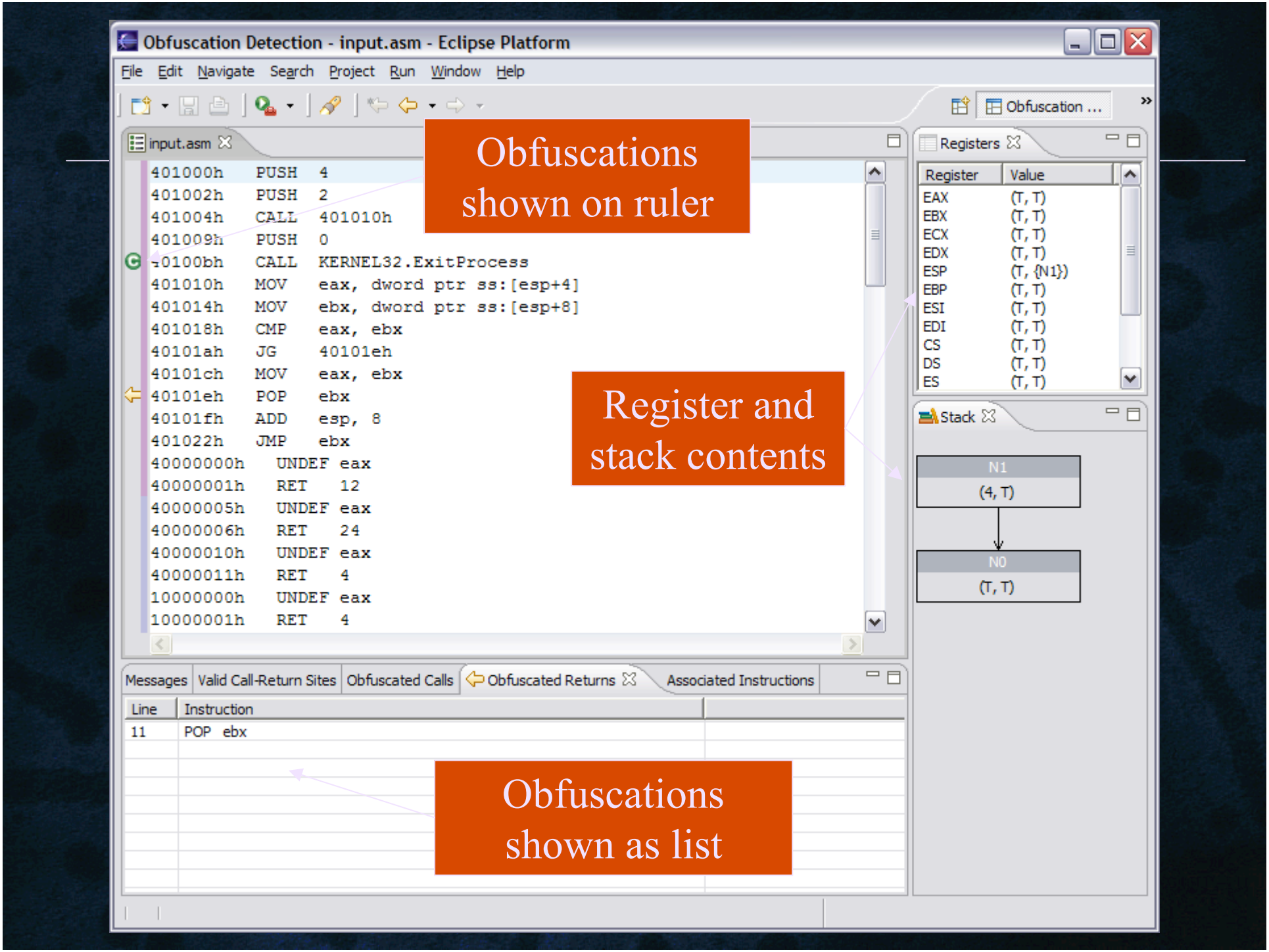
Talk Contents

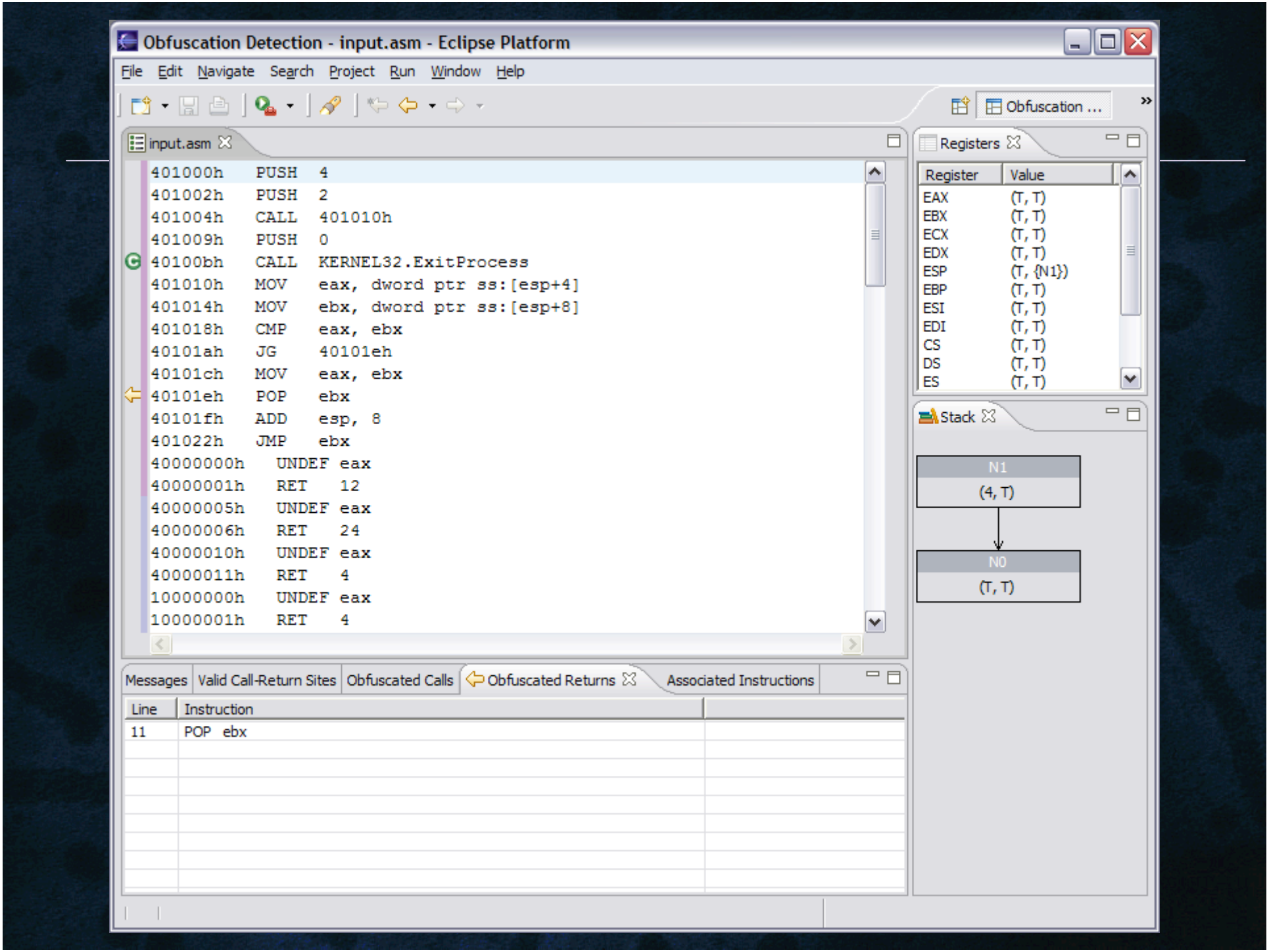
- Motivation
- Previous work
  - VSA (Balakrishnan and Reps, 2004)
  - Abstract Stack Graph (Lakhotia and Kumar 2004)
- VSA+ASG
  - Domain
  - Interpretation example
- Application
  - Detect Call obfuscations
- Implementation
- Future and Conclusions

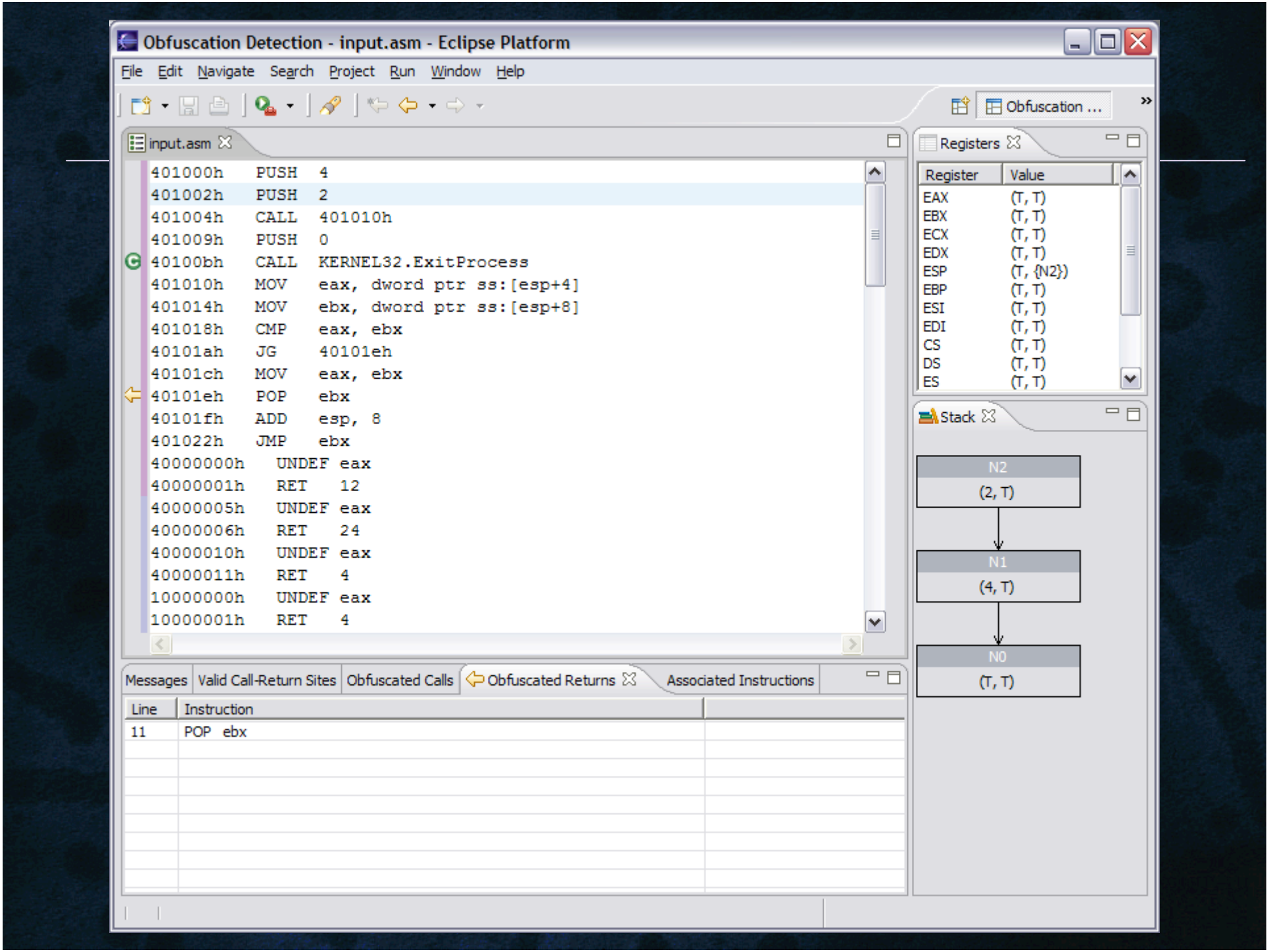
Prototype

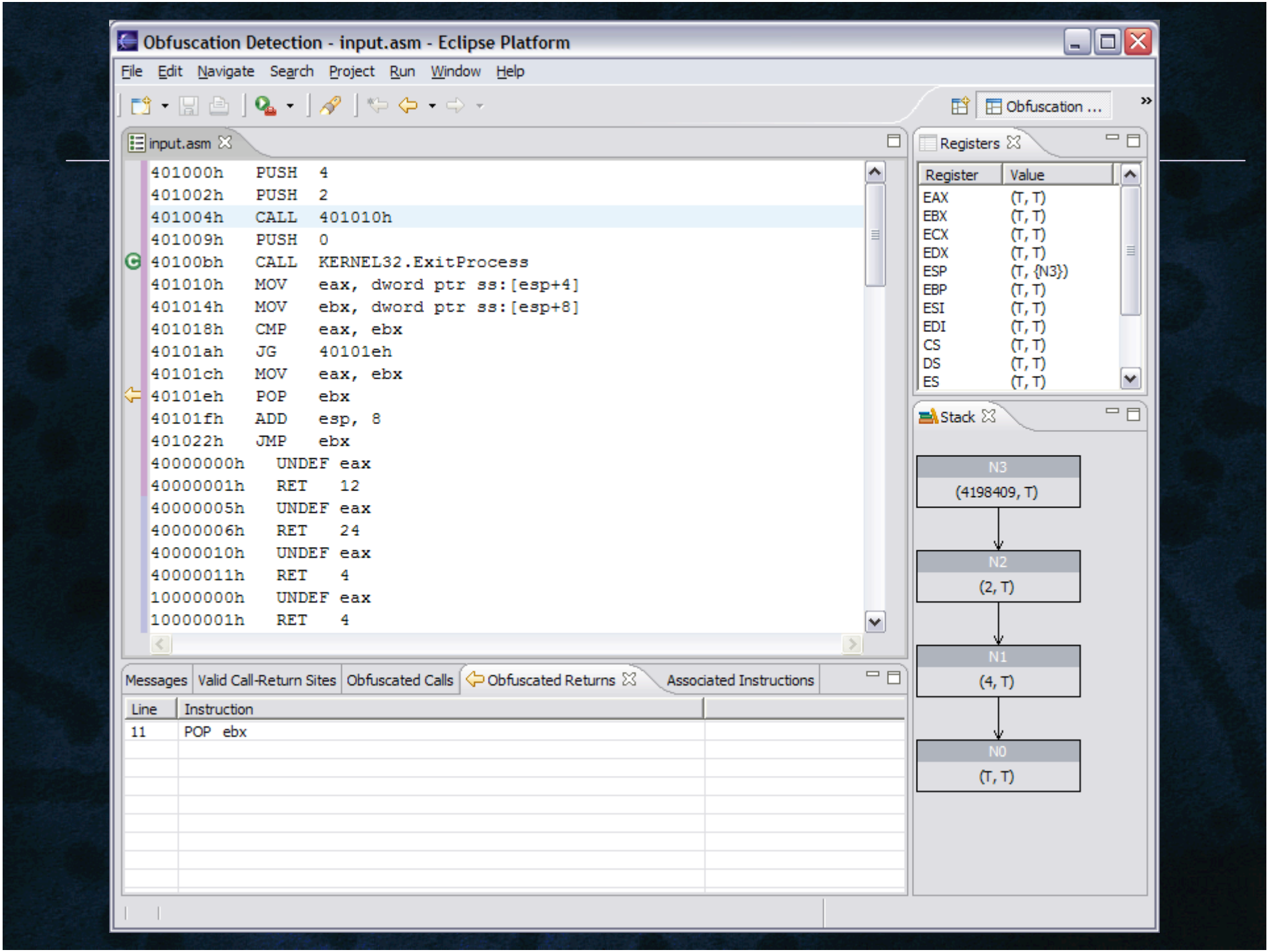
- Implemented on Eclipse platform

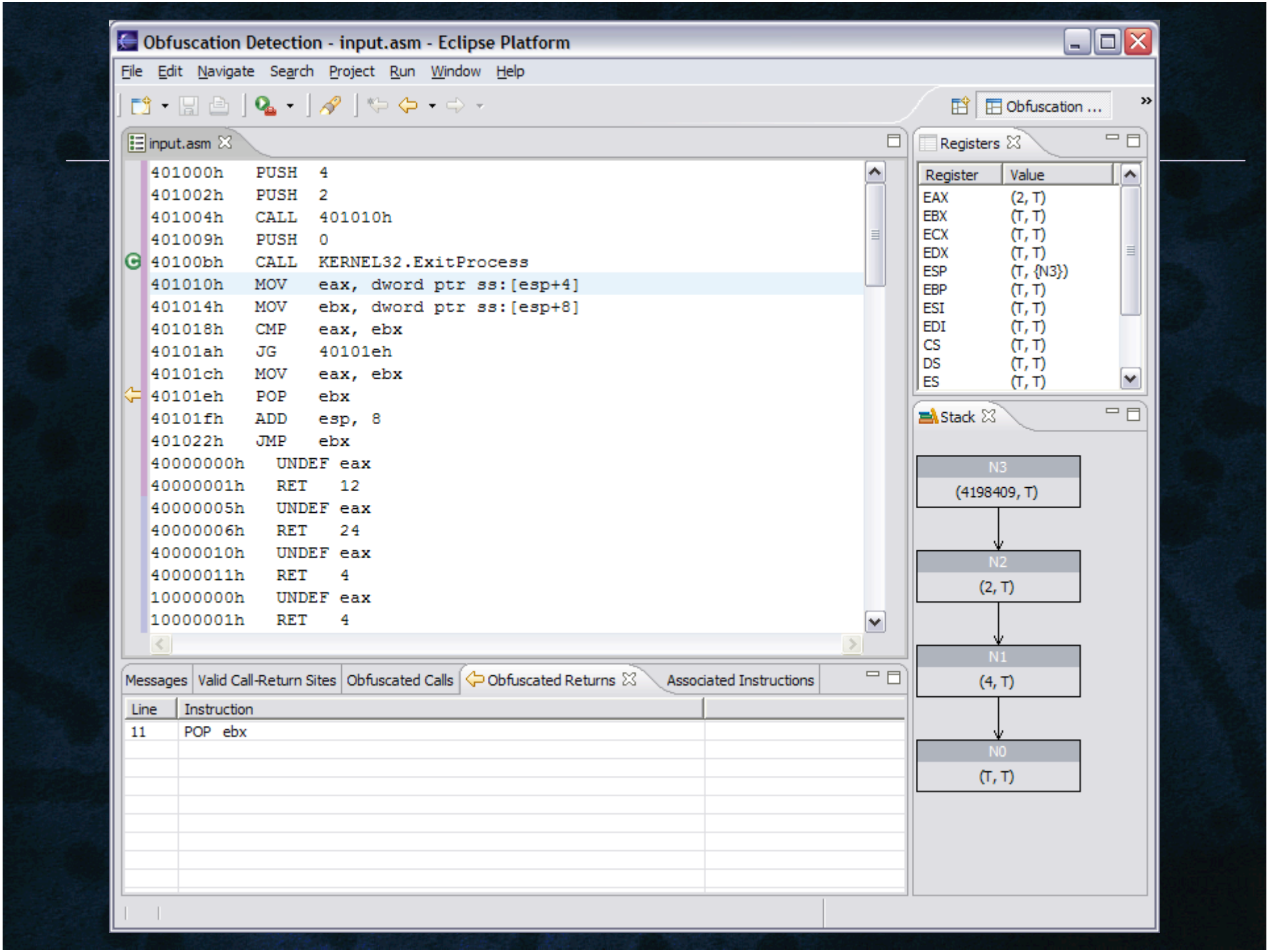


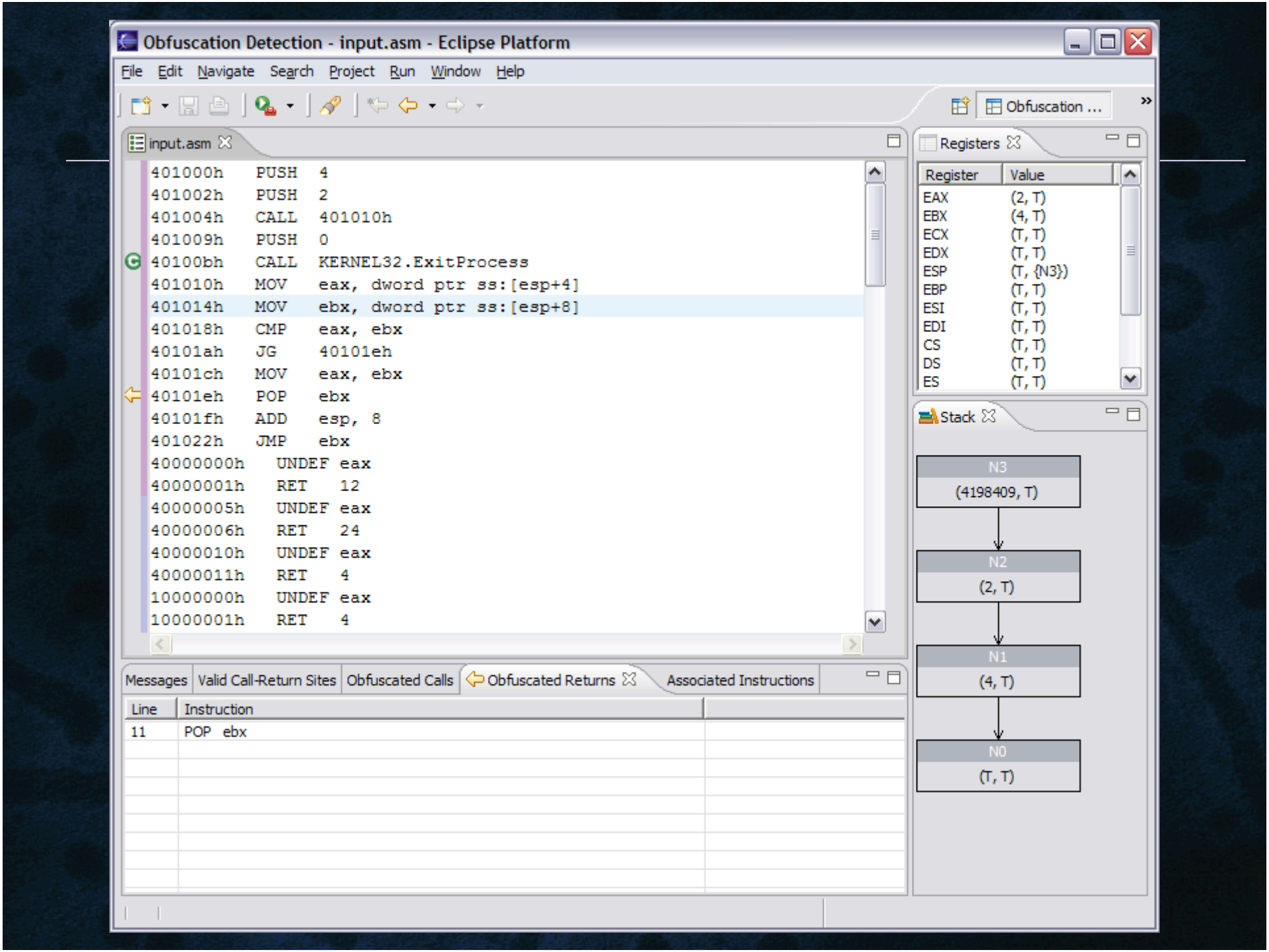

Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Obfuscations shown on ruler
Register and stack contents
Registers
Register Value
EAX (T, T)
EBX (T, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N1})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N1
(4, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx
Obfuscations shown as list

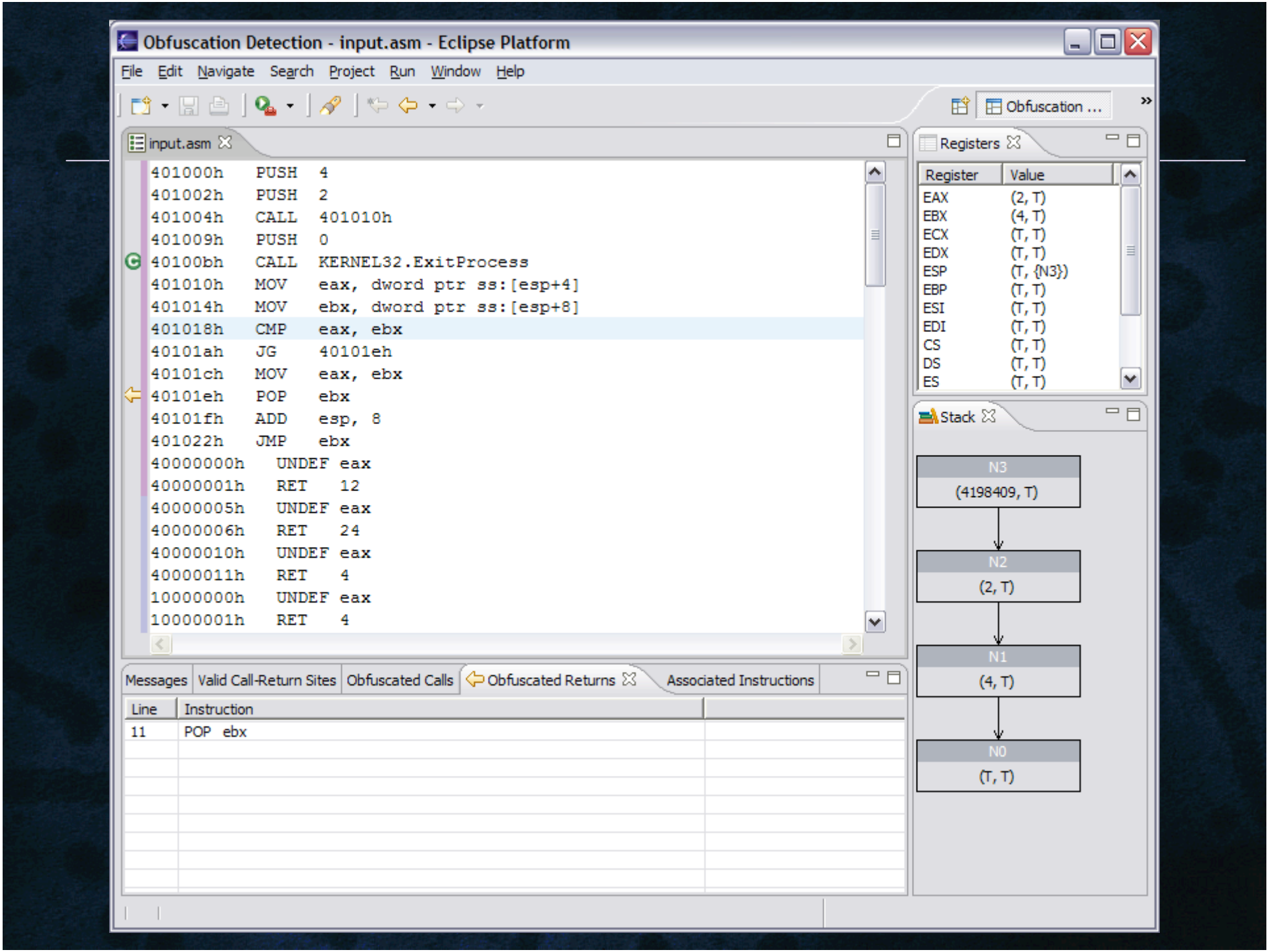

Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (T, T)
EBX (T, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N1})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N1
(4, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx

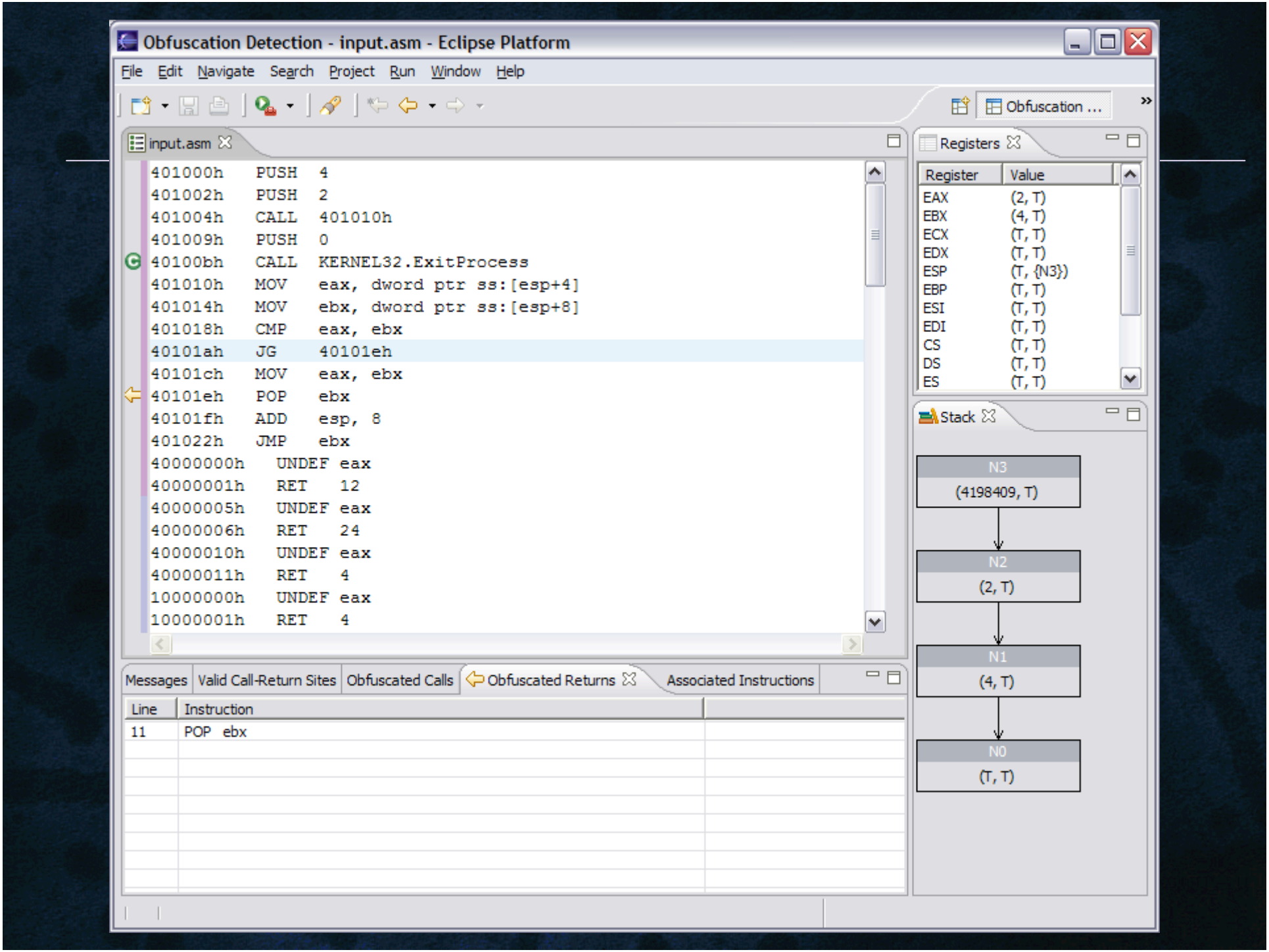

Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (T, T)
EBX (T, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N2})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N2
(2, T)
N1
(4, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx

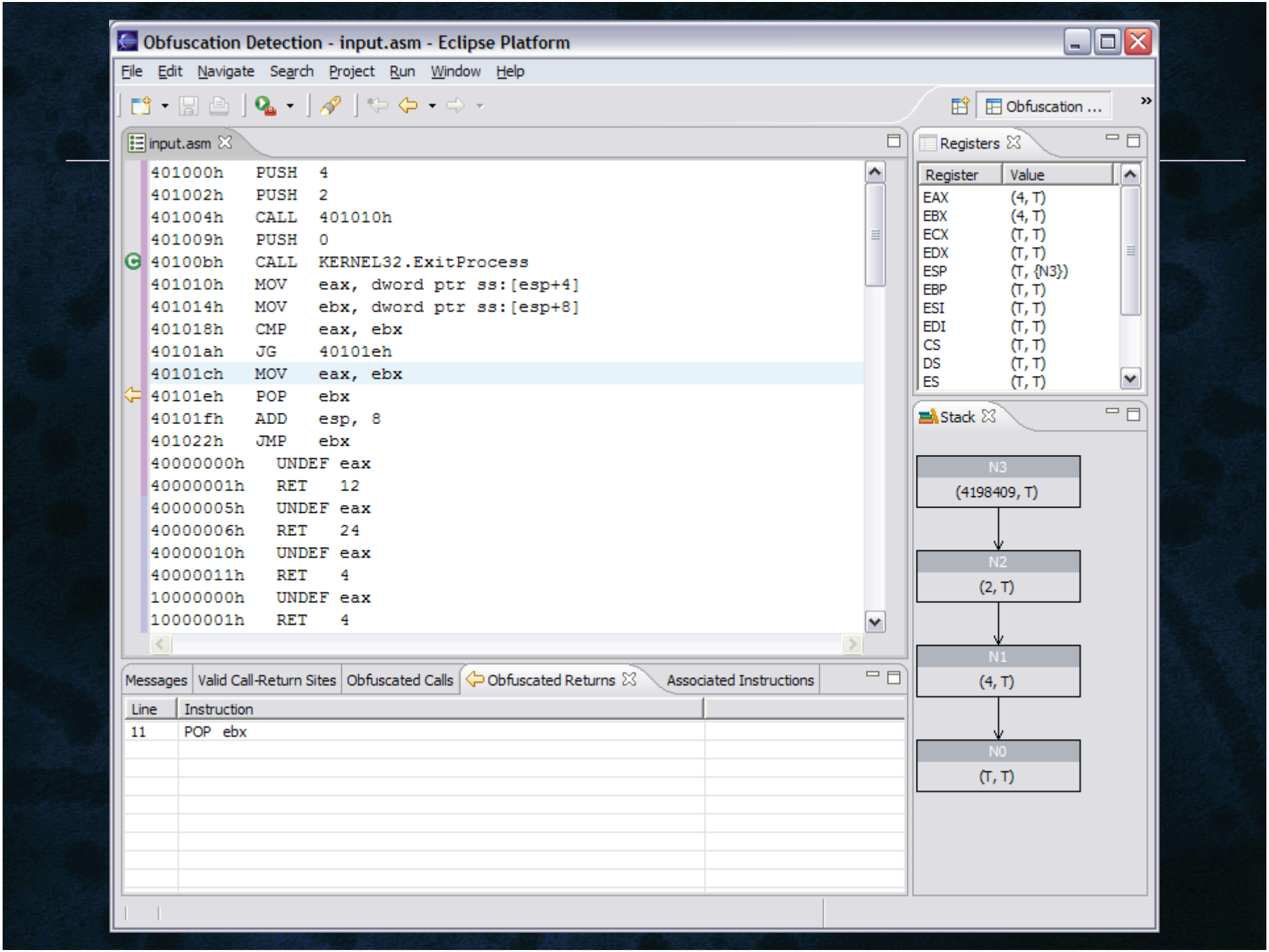

Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (T, T)
EBX (T, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N3})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N3
(4198409, T)
N2
(2, T)
N1
(4, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx

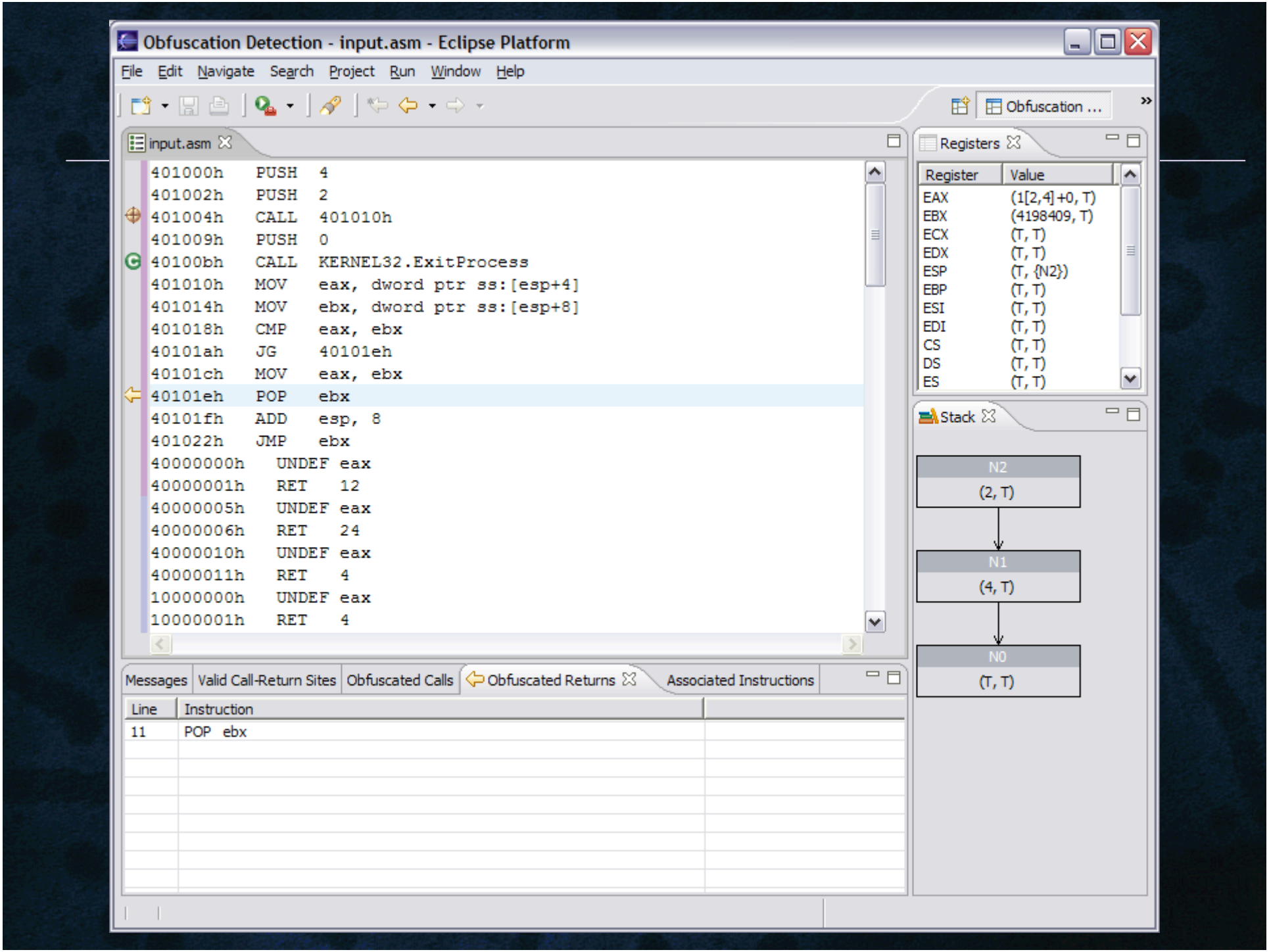

Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (2, T)
EBX (T, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N3})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N3
(4198409, T)
N2
(2, T)
N1
(4, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx

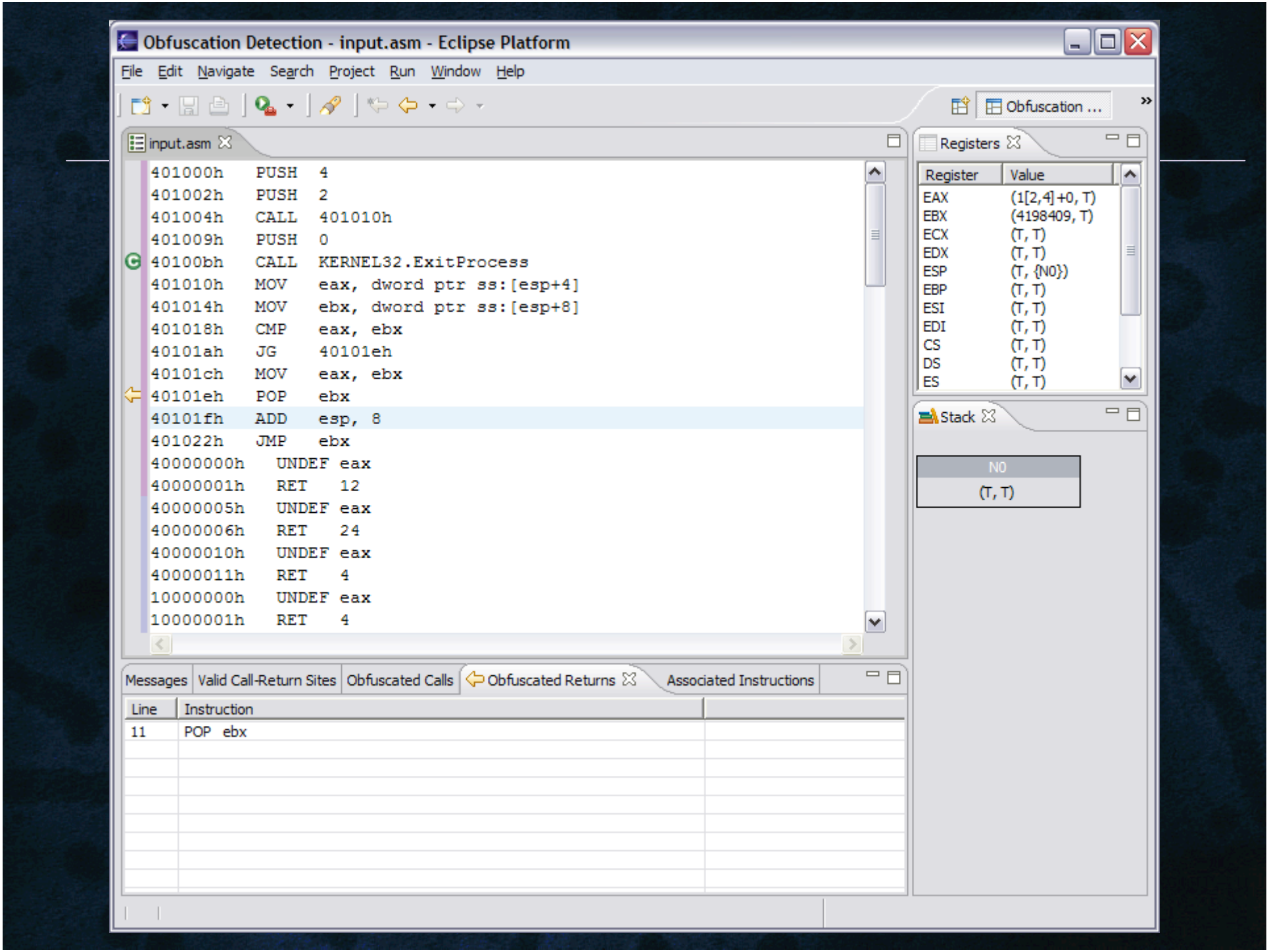

Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (2, T)
EBX (4, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N3})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N3
(4198409, T)
N2
(2, T)
N1
(4, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx

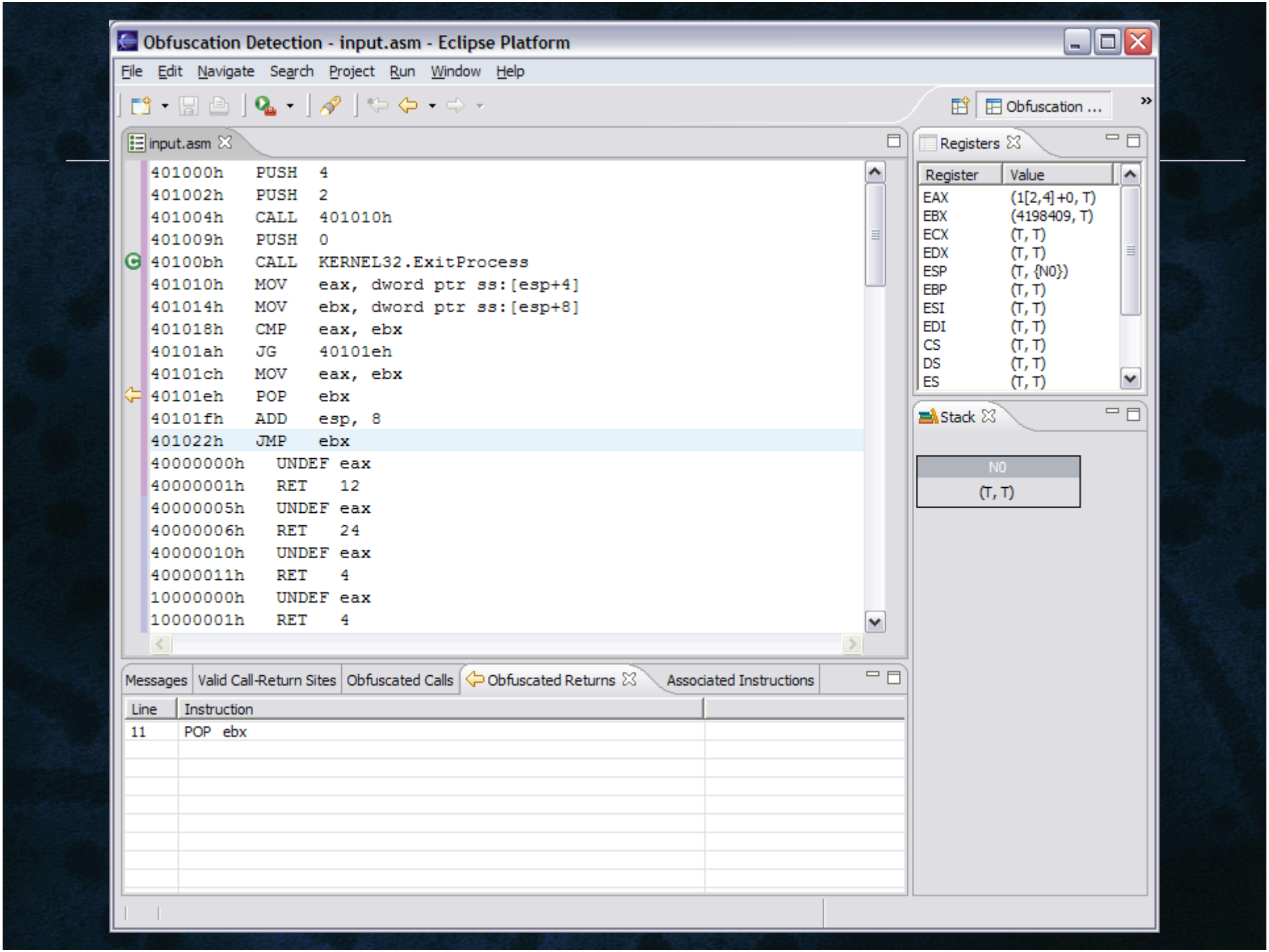

Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (2, T)
EBX (4, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N3})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N3
(4198409, T)
N2
(2, T)
N1
(4, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx

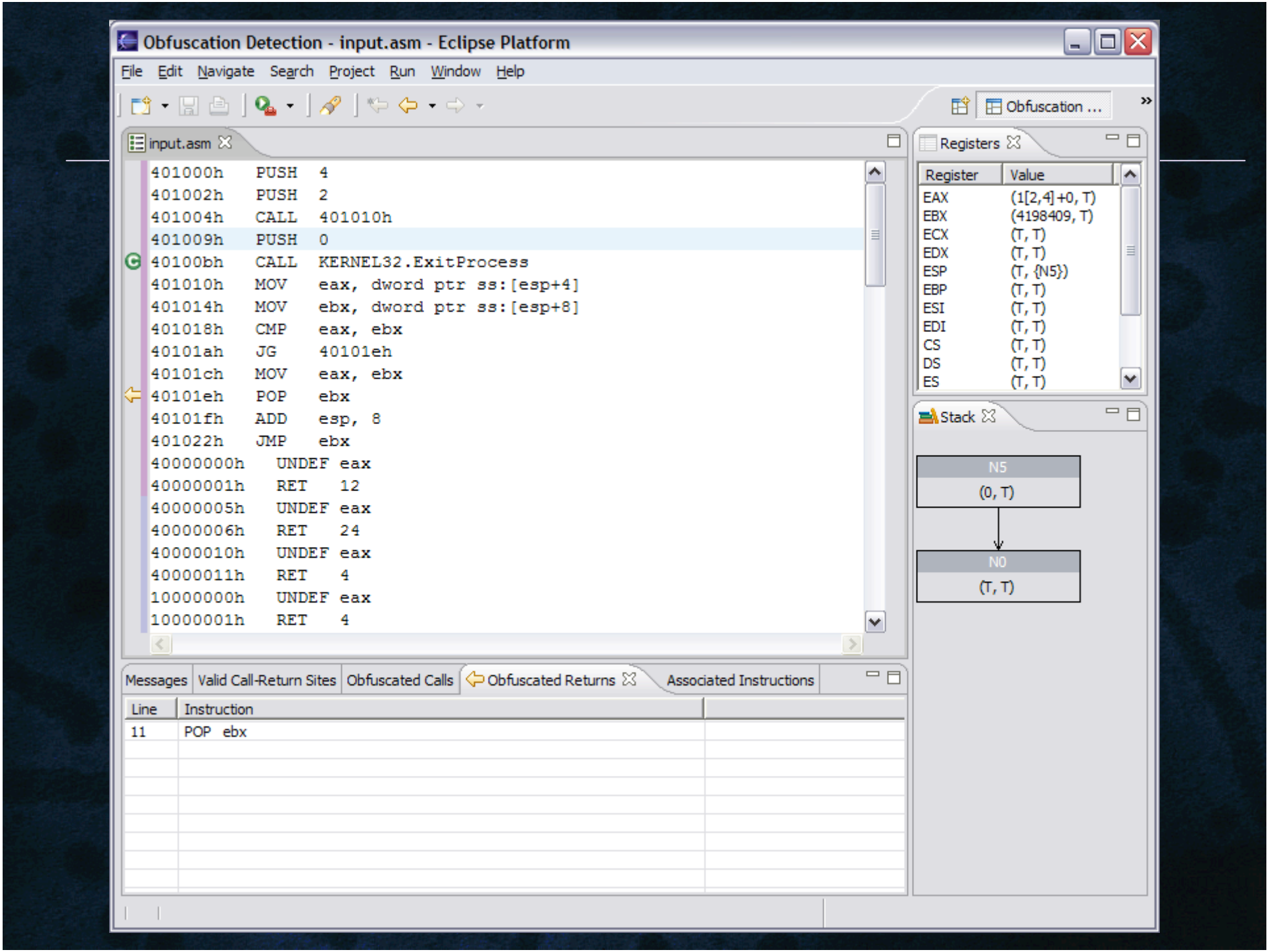

Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (2, T)
EBX (4, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N3})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N3
(4198409, T)
N2
(2, T)
N1
(4, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line
Instruction
11
POP ebx

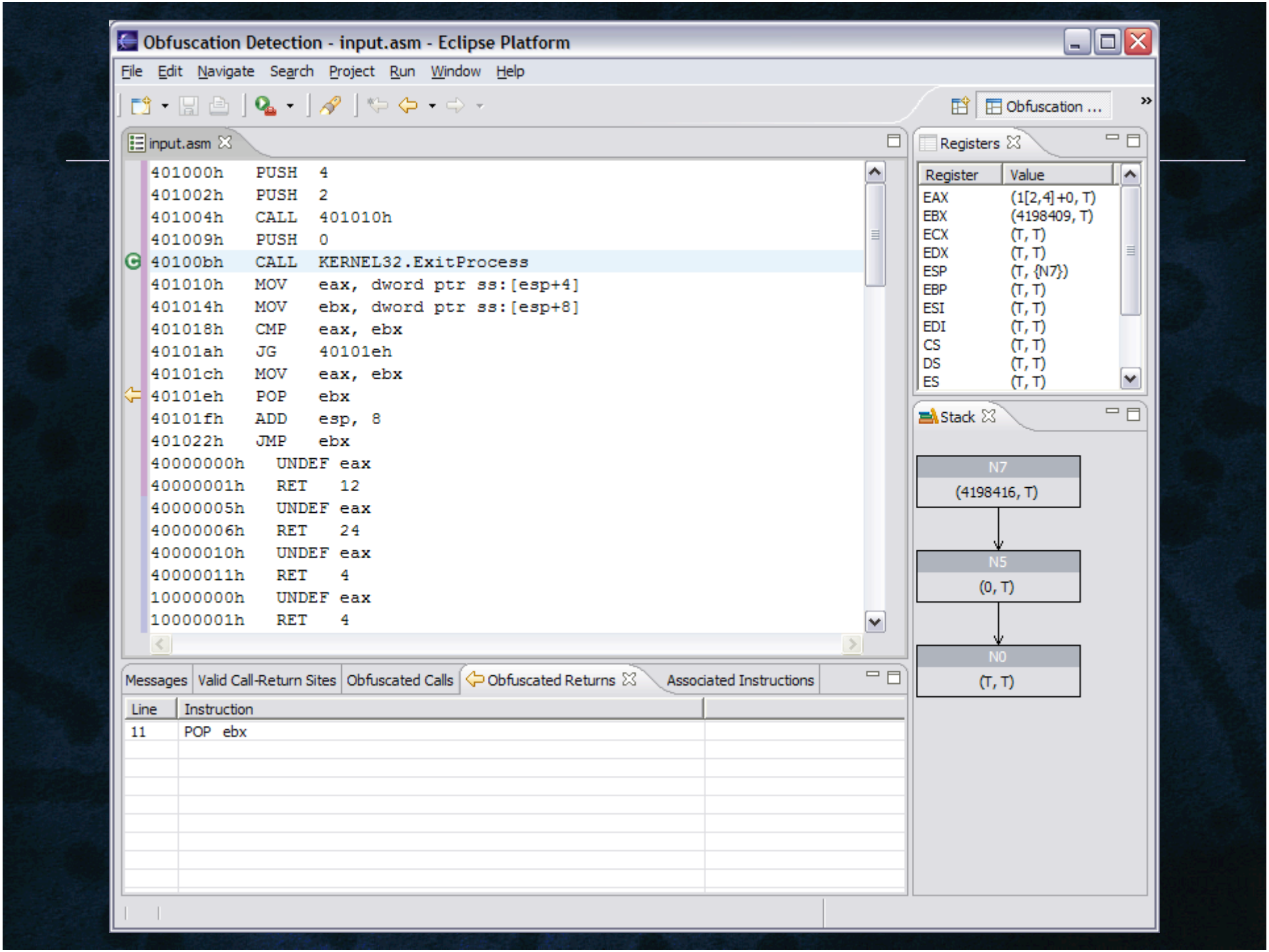

Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (4, T)
EBX (4, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N3})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N3
(4198409, T)
N2
(2, T)
N1
(4, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx

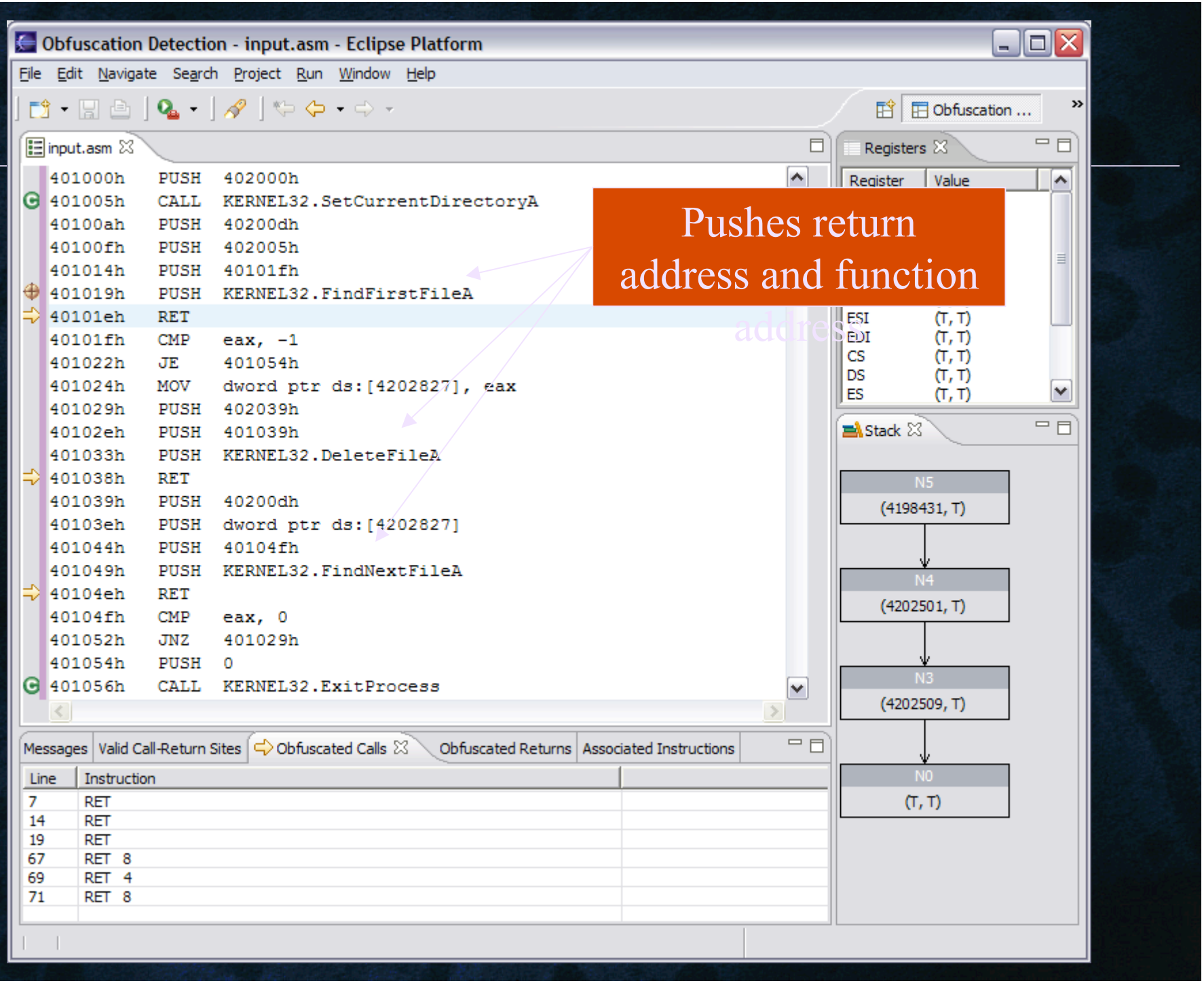

Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (1[2,4]+0, T)
EBX (4198409, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N2})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N2
(2, T)
N1
(4, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx


Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (1[2,4]+0, T)
EBX (4198409, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N0})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx


Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (1[2,4]+0, T)
EBX (4198409, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N0})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx


Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (1[2,4]+0, T)
EBX (4198409, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N5})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N5
(0, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx


Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 4
401002h PUSH 2
401004h CALL 401010h
401009h PUSH 0
40100bh CALL KERNEL32.ExitProcess
401010h MOV eax, dword ptr ss:[esp+4]
401014h MOV ebx, dword ptr ss:[esp+8]
401018h CMP eax, ebx
40101ah JG 40101eh
40101ch MOV eax, ebx
40101eh POP ebx
40101fh ADD esp, 8
401022h JMP ebx
40000000h UNDEF eax
40000001h RET 12
40000005h UNDEF eax
40000006h RET 24
40000010h UNDEF eax
40000011h RET 4
10000000h UNDEF eax
10000001h RET 4
Registers
Register Value
EAX (1[2,4]+0, T)
EBX (4198409, T)
ECX (T, T)
EDX (T, T)
ESP (T, {N7})
EBP (T, T)
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N7
(4198416, T)
N5
(0, T)
N0
(T, T)
Messages
Valid Call-Return Sites
Obfuscated Calls
Obfuscated Returns
Associated Instructions
Line Instruction
11 POP ebx


Obfuscation Detection - input.asm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Obfuscation ...
input.asm
401000h PUSH 402000h
401005h CALL KERNEL32.SetCurrentDirectoryA
40100ah PUSH 40200dh
40100fh PUSH 402005h
401014h PUSH 40101fh
401019h PUSH KERNEL32.FindFirstFileA
40101eh RET
40101fh CMP eax, -1
401022h JE 401054h
401024h MOV dword ptr ds:[4202827], eax
401029h PUSH 402039h
40102eh PUSH 401039h
401033h PUSH KERNEL32.DeleteFileA
401038h RET
401039h PUSH 40200dh
40103eh PUSH dword ptr ds:[4202827]
401044h PUSH 40104fh
401049h PUSH KERNEL32.FindNextFileA
40104eh RET
40104fh CMP eax, 0
401052h JNZ 401029h
401054h PUSH 0
401056h CALL KERNEL32.ExitProcess
Pushes return address and function
Registers
Register Value
ESI (T, T)
EDI (T, T)
CS (T, T)
DS (T, T)
ES (T, T)
Stack
N5
(4198431, T)
N4
(4202501, T)
N3
(4202509, T)
N0
(T, T)
Messages Valid Call-Return Sites Obfuscated Calls Obfuscated Returns Associated Instructions
Line Instruction
7 RET
14 RET
19 RET
67 RET 8
69 RET 4
71 RET 8

Talk Contents

- Motivation
- Previous work
  - VSA (Balakrishnan and Reps, 2004)
  - Abstract Stack Graph (Lakhotia and Kumar 2004)
- VSA+ASG
  - Domain
  - Interpretation example
- Application
  - Detect Call obfuscations
- Implementation
- Future and Conclusions

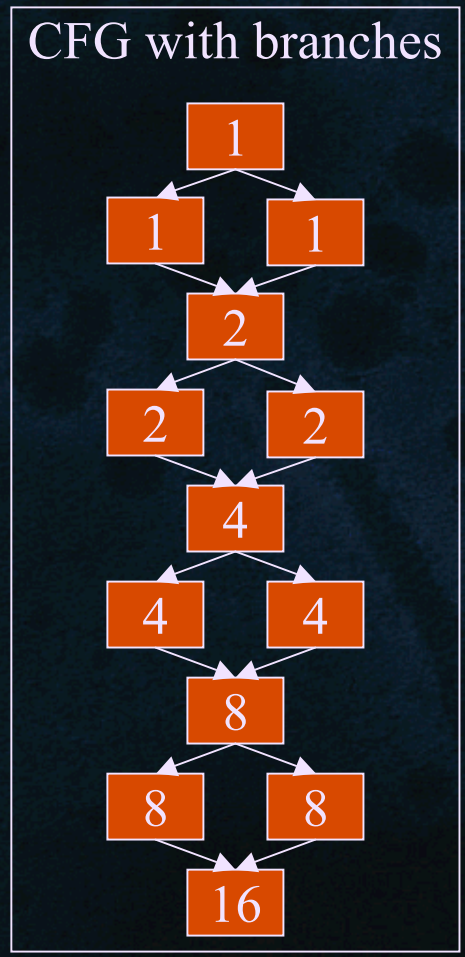
Prototype Limitations

- More Memory Support
  - `mov [eax], ebx`
  - GetModuleHandle/GetProcAddress

```
PUSH offset DLL_NAME     ; “kernel32.dll”
CALL GetModuleHandleA

PUSH offset PROC_NAME    ; “ExitProcess”
PUSH eax
CALL GetProcAddress

PUSH 0
CALL eax
```

Prototype Limitations

- Efficiency
  - Many paths to each instruction
  - More efficient algorithms needed



General Limitations

- More RIC operations
  - Too many undefined RICs
- What about obfuscated call–obfuscated return pairs
  - Not a problem with system calls

General Limitations

- Structure Exception Handling

```
Main:
  PUSH  offset Handler      ; set up exception handler
  PUSH  dword ptr FS:[0]
  MOV   FS:[0], esp

  MOV   edx, 0              ; force an exception
  MOV   eax, 0
  DIV   eax                 ; (edx / eax)

  PUSH  0
  CALL  ExitProcess

Handler:
  ...                       ; real code goes here
```

Conclusion

- Method
  - Combines VSA+ASG
  - Interpret using abstract values
  - Apply rules to abstract stack graph to detect obfuscations
- Results
  - Can handle more instructions (`mov esp, eax`)
  - Can handle indirect jumps/calls
  - Anticipate support for GetModuleHandle/GetProcAddress
- Prototype performed as expected, though more work is needed