segment data

aici DD here ; aici := offset(here)

segment code

. . .

mov eax, [aici] ; mov eax, dword ptr DS:[aici] ; = EAX:=offset(here)

mov ebx, aici ; EBX := offset(aici)

jmp [aici] ; salt determinat de continutul variabilei aici si efectuat la adresa (offsetul) reprezentat de acest continut – mai exact, saltul va fi echivalent cu jmp la offset(here) deci echivalenta ca efect cu JMP here

jmp here ; salt la eticheta here

jmp eax ; salt la valoarea de adresa offset continuta de registrul EAX. Cum EAX = offset(here) saltul va fi echivalent cu jmp la offset(here) deci echivalenta ca efect cu JMP here

jmp [ebx] ; salt determinat de continutul variabilei DS:[ebx] si efectuat la adresa (offsetul) reprezentat de acest continut – EBX = offset (aici), deci DS:[EBX] = offset(here), deci saltul va fi echivalent cu jmp la offset(here) deci echivalenta ca efect cu JMP here

jmp [ebp] ; JMP DWORD PTR SS:[EBP]

jmp [EBP\*4] ; JMP DWORD PTR DS:[EBP]

...............................................................

here:

mov ecx, 0ffh

........

jmp [aici]

**- EAX**

offset here

offset here

**- aici**

[ebx]

**- EBX**

offset aici

jmp [ebx]

jmp eax

here:

segment data (la prima parcurgere a exemplului ignorati zonele marcate cu galben !!)

aici DD here ;echivalent cu aici := offsetul etichetei here din segmentul de cod

segment code

. . .

mov eax, [aici] ;se încarcă în EAX conţinutul variabilei aici (adică deplasamentul lui here în cadrul segmentului code *–* echivalent deci ca efect cu: **mov eax, here**)

mov ebx, aici ;se încarcă în EBX deplasamentul lui aici în cadrul segmentului data

; (00401000h)

jmp [aici] ;salt la adresa desemnată de valoarea variabilei aici(care este adresa lui here), deci instrucţiune echivalentă cu jmp here ; ce face **jmp aici** ??? – acelasi lucru ca si jmp ebx ! **salt la CS:EIP cu EIP=offset (aici) din SEGMENT DATA (00401000h)** ; salt la niste instr. care merg pana la primul access violation

jmp here ;salt la adresa lui here (sau, echivalent, salt la eticheta here); **jmp [here]** ?? – JMP DWORD PTR DS:[00402014] – cel mai probabil Access violation....

jmp eax ;salt la adresa conţinută în EAX (adresare registru în mod direct), adică la here

; ce face prin comparatie **jmp [eax]** ??? JMP DWORD PTR DS:[EAX] – cel mai

; probabil Access violation....

jmp [ebx] ;salt la adresa conţinută în locaţia de memorie a cărei adresă este conţinută în ;EBX (adresare registru în mod indirect - singura situaţie de apel indirect din ;acest exemplu) – ce face prin comparatie **jmp ebx** ??? - **salt la CS:EIP cu EIP=offset (aici) din SEGMENT DATA (00401000h)** ; salt la niste instr. care merg pana la primul access violation

;în EBX se află adresa variabilei aici, deci se accesează conţinutul acestei variabile. În această locaţie de memorie se găseşte offset-ul etichetei here, deci se va efectua saltul la adresa here *-*  **ca urmare, ultimele 4 instrucţiuni de mai sus sunt toate echivalente cu jmp here**

jmp [ebp] ; JMP DWORD PTR SS:[EBP]

...............................................................

here:

mov ecx, 0ffh

|  |  |
| --- | --- |
| **MOVZX d, s** | încarcă în d (REGISTRU !), de dimensiune mai mare decât s (registru sau operand din memorie!), continutul lui s **fară semn** |
| **MOVSX d, s** | încarcă în d (REGISTRU !), de dimensiune mai mare decât s (registru sau operand din memorie!), continutul lui s **cu semn** |

MOV ah, 0c8h ; AH = C8h

MOVSX ebx, ah ; EBX = FFFF FFC8h

MOVZX edx, ah ; EDX = 0000 00C8h

MOVSX ax,[v] ; MOVSX ax, byte ptr DS:[offset v]

MOVZX eax, [v] ; syntax error – operation size not specified !

MOVZX eax, byte/word [v] ; ok !

MOVZX eax, dword [v] ; syntax error !

Mov bh,1 ; BH = 1

Movsx ax,bh ; AX = 1

Atention ! These are NOT syntactically accepted:

CBD CWDE EBX, BX MOVSX EAX, [v]

CWB CWD EDX,AX MOVZX EAX, [EBX]

CDW MOVZX AX, BX MOVSX dword [EBX], AH

CDB !!! MOVSX EAX, -1 CBW BL

(super-înghesuire!! ☺)

A[7] = \*(A+7) = \*(7+A) = 7[A] !!! – MERGE !!!!

CALL = salvarea adresei de revenire pe stiva + JMP la eticheta

RET = extragerea adresei de revenire de pe stiva + JMP la acea adresa de revenire

D < S ??? echiv. D-S < 0 ????

Dinnou:

Mov eax, 89

Jmp maideparte

Resd 1000h; Distanta intre LOOP si eticheta Dinnou este > 127 bytes deci nu este short jump !

Maideparte:

Mov ebx, 17

Loop dinnou dec ecx

jnz dinnou

Et1:

Mov ebx, 213

…….

Resd 100h

……..

Js et1

Alternativa este:

Et1:

Mov ebx, 213

…….

……..

Jns Maideparte

Jmp et1

Maideparte:

În TASM și MASM condiția de salt short (adica maxim 127 de octeți distanță) este impusă atât la nivel de instrucțiuni de tip LOOP, cât și la nivel de instrucțiuni de salt condiționat.

În NASM restricția este valabilă doar pentru instrucțiuni de tip LOOP, instrucțiunile de salt condiționat nemaifiind supuse acestei restricții