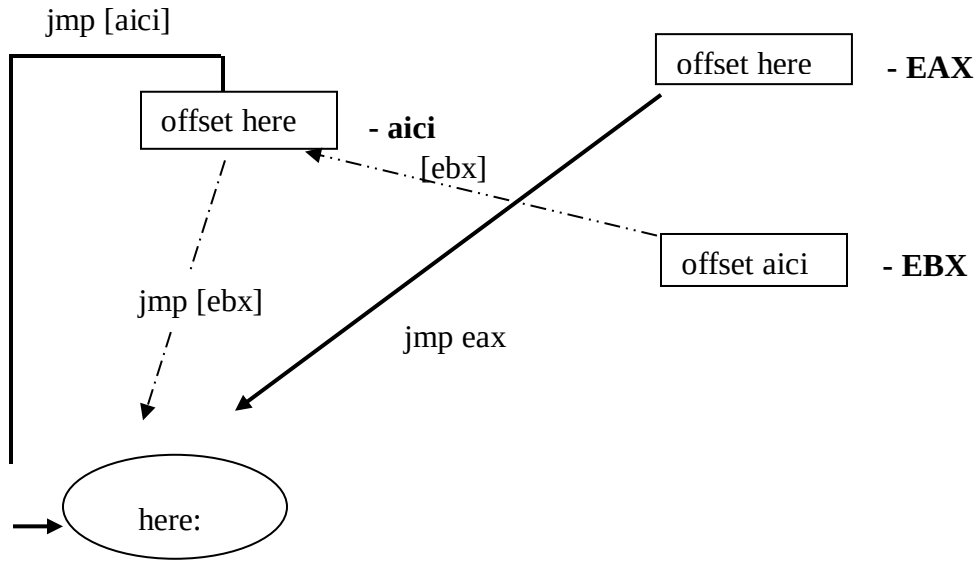
```
segment data
        aici   DD  here  ; aici :=offset(here)

segment code
        .       .       .
        mov  eax, [aici]   ;  EAX := offset(here)

        mov  ebx, aici ; EBX := offset (aici) = 0 (computed by the assembler) ; = 00401000h

         jmp   [aici]   ; JMP to the offset taken from the contents of aici which is the offset of here... similar with
                        JMP HERE

        jmp   here      ; direct access of the target code label

        jmp eax         ; JMP to the offset taken as the contents of EAX, which is the offset of here... similar with
JMP HERE

       jmp   [ebx]      ; jump to the address stored in the memory location having the address THE CONTENTS OF
                        EBX (indirect register access)  ; this will be variable aici from which we take offset(here) so
                        the jump will also be made to offset(here), so similar to JMP HERE

        jmp   [ebp]     ;  JMP DWORD PTR SS:[EBP] – don't know exactly what does... possible Memory access
                        violation...
.....................................................
        here:
                mov ecx, 0ffh
```



---

segment data (ignore yellow highlighted text at the first parsing !!)

```
        aici   DD  here          ;equiv. with  aici := offsetul of label here from code segment

segment code
        .       .       .
        mov  eax, [aici]        ;we load EAX with the contents of variable aici (that is the offset of here inside the
                                  code segment – same effect as  mov eax, here)

        mov  ebx, aici          ;we load EBX with the offset of aici inside the data segment
                                ; (probable 00401000h)

      jmp   [aici]      ;jump to the address indicated by the value of variable aici (which is the address of  here), so
                          this is an instruction equiv with jmp here  ; what does in contrast jmp aici ??? - the
                          same thing as jmp ebx ! jump to CS:EIP with EIP=offset (aici) from SEGMENT
                          DATA (00401000h) ; jump to some instructions going until the first „access
                          violation”

        jmp   here      ;jump to the address of here (or, equiv, jump to label here); jmp [here] ?? – JMP
        DWORD PTR DS:[00402014] – most probably „Access violation”....


        jmp  eax        ;jump to the address indicated by the contents of EAX (accessing register in direct mode), that
is to label here ; in contrast, what does jmp [eax] ???  JMP DWORD PTR DS:[EAX] – most
                                                          probably „Access violation”....

        jmp   [ebx]     ;jump to the address stored in the memory location having the address the contents of EBX
                          (indirect register access – the only indirect access from this example) – what does in contrast
                          jmp ebx ???  - jump to CS:EIP with EIP=offset (aici) from the  SEGMENT DATA
                          (00401000h) ; jump to some instructions going until the first „access violation” or
                          it may work if label start from code segment is really at offset 0 (00402000h)
;in EBX we have the address of variable aici, so the contents of this variable will be accessed. In this memory location
we find the offset of label here, so a jump to address here will be performed - consequently, the last 4 instructions
from above are all equivalent to jmp here

        jmp   [ebp]   ;  JMP DWORD PTR SS:[EBP]
.....................................................................
        here:
                mov ecx, 0ffh
```

```
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!! ☺)

```
Dinnou:
   Mov eax, 89
   Jmp maideparte

   Resd 1000h

   Maideparte:
   Mov ebx, 17

Loop dinnou
```

```
mov  eax, [aici]    ; mov eax, dword ptr DS:[aici]
mov eax, [FS:aici]

XLAT
ES XLAT
```

```
Dinnou:
   Mov eax, 89
   Jmp maideparte

      Resd 1000h; The distance between LOOP and the label Dinnou is > 127 bytes so it is not a short jump !

   Maideparte:
   Mov ebx, 17

Loop dinnou                dec ecx
                           jnz dinnou



Et1:
     Mov ebx, 213
     …….
     Resd 100h
     ……..
Js et1
```

The alternative is:

```
Et1:
     Mov ebx, 213
     …….

     ……..
     Jns Maideparte
Jmp et1

Maideparte:
```

In TASM and MASM the short jump condition (ie maximum 127 bytes distance) is imposed both at the level of LOOP type instructions and at the level of conditional jump instructions.

In NASM the restriction is valid only for LOOP type instructions, the conditional jump instructions are no longer subject to this restriction.