

EXEMPLU DE UTILIZARE ȘI EXPLOATARE A MEMORIEI INTERNE

Să se determine harta memoriei pentru datele de mai jos și offset-urile (deplasările) șirurilor

```
segm_stiva SEGMENT
    stiva    DB 128 DUP (?)
segm_stiva ENDS
```

```
segm_date SEGMENT
    sir_date  DB    7,10,11, ?
    sir_date2 DW    7,10,11, ?
    sir_car   DB    'ABC 123'
    car       DB    '13'
    nr        DB    13
    nr_mare  DD    1abcdefh
segm_date ENDS
```

```
segm_cod SEGMENT
    ASSUME CS:segm_cod, DS:segm_date, SS:segm_stiva
```

```
et:    MOV AX, segm_date
        MOV DS, AX
```

```
    MOV SI, OFFSET sir_date      ; SI = 00 00
    MOV AL, [SI]                 ; AX = ....07
    MOV BL, [SI+1]               ; BX = ....0A
    MOV CL, [SI+2]               ; CX = ....0B
    MOV [SI+3], AL               ; valoarea 07 se mută la adresa 0003
```

```
    MOV DI, OFFSET sir_date2    ; DI = 00 04
    MOV AX, [DI]                 ; AX = 00 07
    MOV BX, [DI+2]               ; BX = 00 0A
    MOV CX, [DI+4]               ; CX = 00 0B
    MOV [DI+6], AX               ; valoarea 00 07 se mută la adresa
```

000A

```
    MOV SI, OFFSET sir_car      ; SI = 00 0C
    MOV AL, [SI]                 ; AX = ....41
    MOV BL, [SI+1]               ; BX = ....42
    MOV CL, [SI+2]               ; CX = ....43
```

```
    MOV DI, OFFSET car          ; DI = 00 13
    MOV AL, car                  ; AX = .... 31
    MOV AH, nr                   ; AX = 0D ...
```

```
    MOV SI, OFFSET nr_mare      ; SI = 00 16
    MOV AX, WORD PTR [nr_mare]  ; AX = CD EF
    MOV BX, WORD PTR [nr_mare+2] ; BX = 01 AB
```

```
MOV AX, 4C00H
INT 21H
```

```
segm_cod ENDS
```

```
END et
```

```

CPU 80486
cs:0000 B8B04E mov ax,4EB0
cs:0003 8ED8 mov ds,ax
cs:0005 BE0000 mov si,0000
cs:0008 8A04 mov al,[si]
cs:000A 8A5C01 mov bl,[si+01]
cs:000D 8A4C02 mov cl,[si+02]
cs:0010 884403 mov [si+03],al
cs:0013 BF0400 mov di,0004
cs:0016 8B05 mov ax,[di]
cs:0018 8B5D02 mov bx,[di+02]
cs:001B 8B4D04 mov cx,[di+04]
cs:001E 894506 mov [di+06],ax
cs:0021 BE0C00 mov si,000C
cs:0024 8A04 mov al,[si]
cs:0026 8A5C01 mov bl,[si+01]

ax 4E07 c=0
bx 000A z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0000 d=0
ds 4EB0
es 4E98
ss 4EA8
cs 4EB2
ip 000D

[ ]=Dump 2=[ ]
ds:0000 07 0A 0B 00 07 00 0A 00 00 00 00 00 00 00
ds:0008 0B 00 00 00 41 42 43 20 08 ABC
ds:0010 31 32 33 31 33 0D EF CD 12313Fn=
ds:0018 AB 01 00 00 00 00 00 00 00 00 00 00 00 00
ss:0008 0000
ss:0006 0000
ss:0004 0000
ss:0002 0000
ss:0000 0000

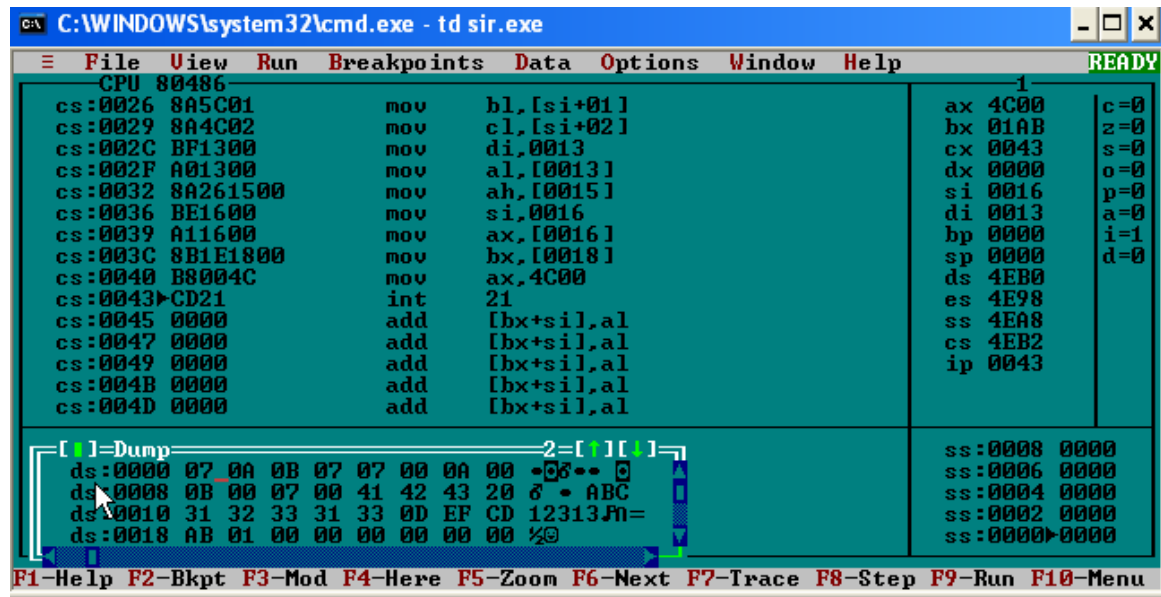
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

Harta memoriei la începutul programului:

<i>sir_date</i>	<i>sir_date2</i>	<i>sir_car</i>	<i>car</i>	<i>nr</i>
07 0A 0B 00	07 00 0A 00 0B 00 00 00	41 42 43 20 31 32 33	31 33	0D
EF CD AB 01				
<i>nr_mare</i>				

OBS.: În memorie, numerele la nivel de B (byte) și caracterele (șirurile de caractere) sunt reprezentate în ordinea normală, iar numere la nivel de W (word) și DD (double word) sunt reprezentate invers (se inversează byte-s între ei) – reprezentarea „little endian”



Harta memoriei la sfârșitul programului:

<i>sir_date</i>	<i>sir_date2</i>	<i>sir_car</i>	<i>car</i>	<i>nr</i>
-----	-----	-----	-----	---
07 0A 0B 07	07 00 0A 00 0B 00 07 00	41 42 43 20 31 32 33	31 33	0D
EF CD AB 01				

<i>nr_mare</i>				