18. Se da un sir de dublucuvinte. Sa se ordoneze crescator sirul cuvintelor superioare ale acestor dublucuvinte. Cuvintele inferioare raman neschimbate.

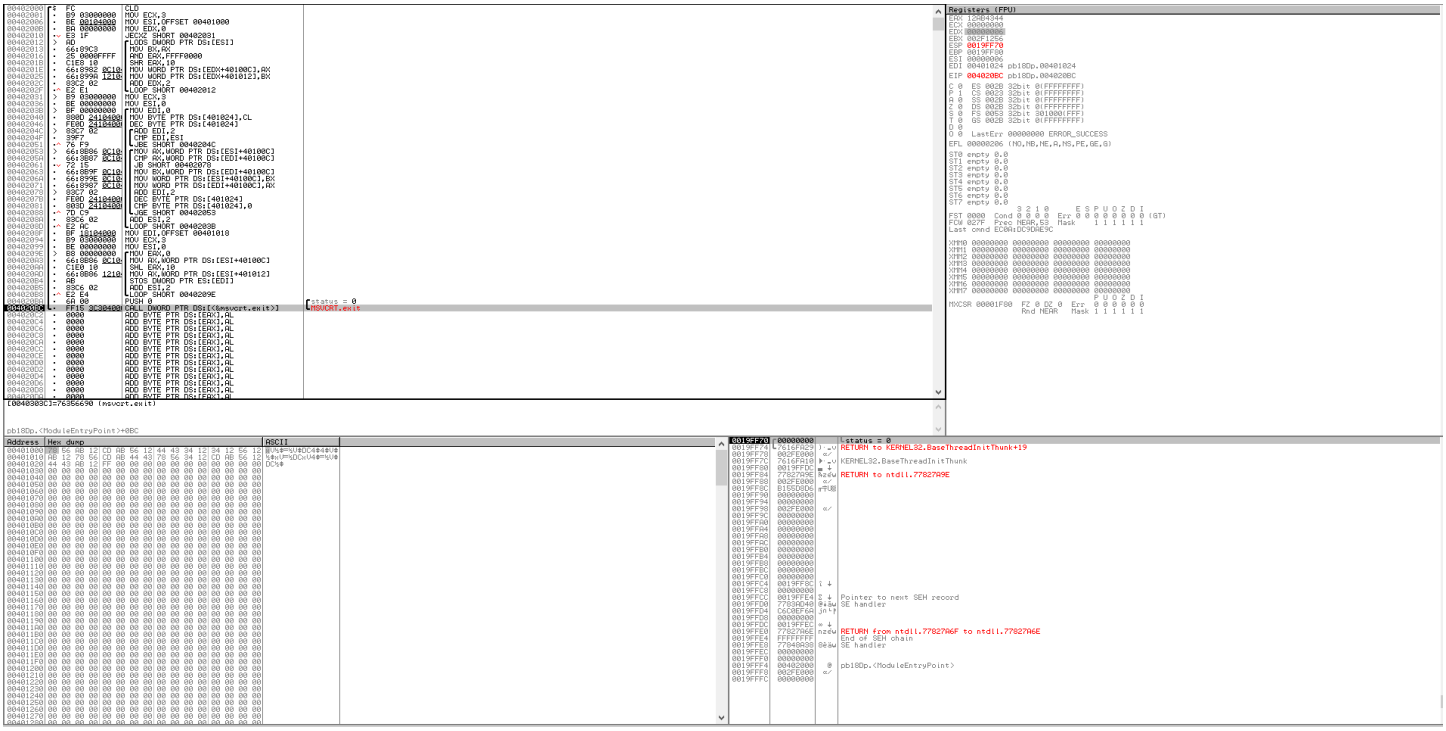
Exemplu:

dandu-se:

```
sir DD 12AB5678h, 1256ABCDh, 12344344h
```

rezultatul va fi

```
12345678h, 1256ABCDh, 12AB4344h.
```



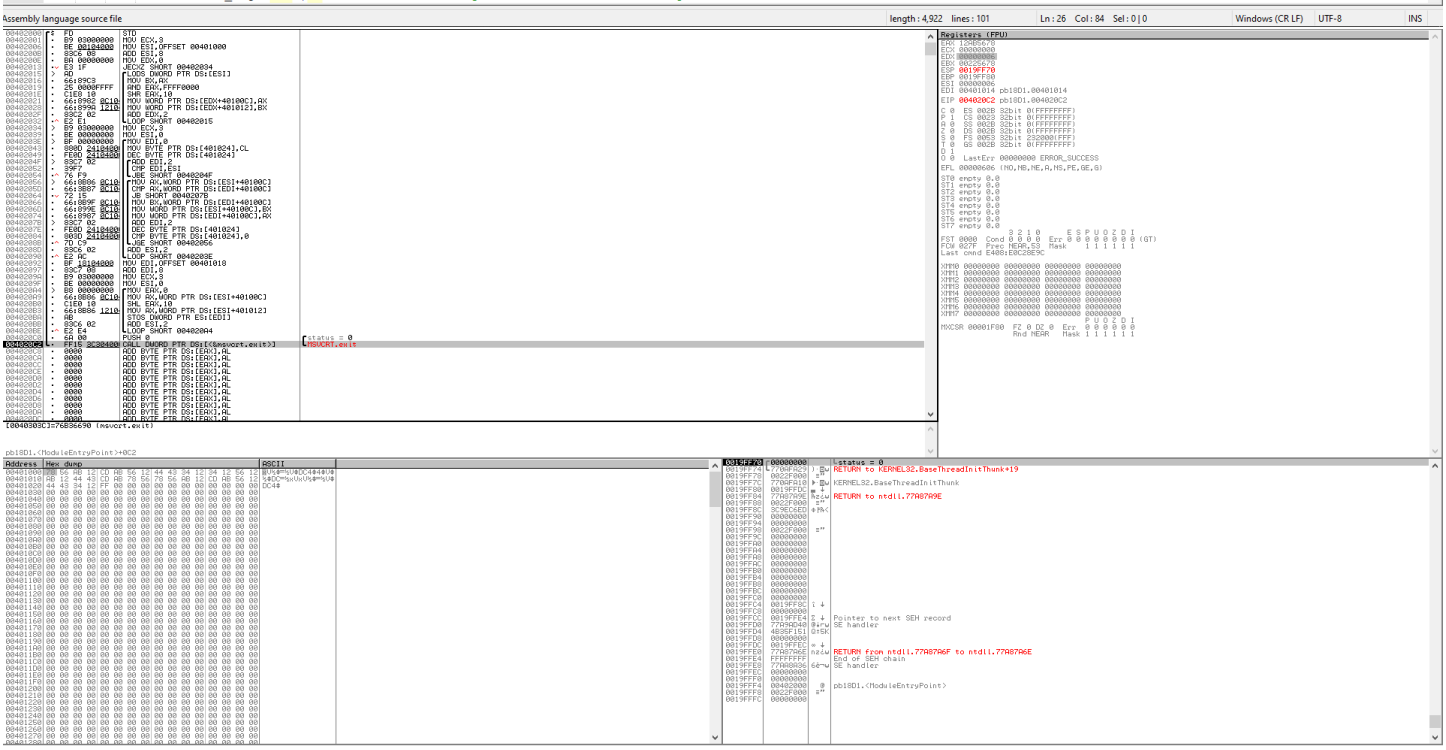
```
31          mov BX,AX ;aducem in BX partea low a dublucuvantului
32
33          and EAX,0FFFF0000h ;izolam cuvantul high din EAX
34          shr EAX,16 ;aducem dublucuvantul high in locul dublucuvantului low
35          ;acum avem BX si in AX cele doua componente ale numarului cu care vom opera mai departe
36
37          mov [sir_high+EDX],AX  ;punem partea high, stocata in AX in sir_high
38          mov [sir_low+EDX] ,BX  ;punem partea low, stocata in BX in sir_low
39          add EDX,2              ;incrementam cu 2 indexul sirurilor deoarece lucram cu word-uri
40
41      loop st_loop
42      sfarsit:
43      ;vom proiecta un algoritm de sortare pentru sir_high
44      mov ECX,l ;punem in registrul contor, lungimea sirului
45      mov ESI,0 ;punem in ESI 0, pentru a parcurge sirul cu el
46
47      bg_loop:
48
49          mov EDI,0          ;initializam EDI cu 0, folosindul ca index pentru a realiza sortare
50          mov byte[cont],CL  ; folosim cont pe post de variabila contor, punem in ea nr ramas de pasi
51          dec byte[cont]     ; decrementam cu o unitate aferenta cu inceperea de la j+1 a for-ului din sortarea prin selectie
52
53          stra:
54          add EDI,2          ;il aducem prin aceasta structura pe EDI la valoarea lui ESI
55          cmp EDI,ESI
56          jbe stra
57
58          slp:
59
60              mov AX,[sir_high+ESI]       ;preluam in AX elementul din sirul high (elementul de pe pozitia i)
61              cmp AX,word[sir_high+EDI]   ;comparam cu elementul de pe pozitia j
62
63              jb mic
64
65              mov BX,[sir_high+EDI]       ;in cazul in care am gasit doua elemente ce trebuie interschimbate
66              mov [sir_high+ESI],BX       ;realizam swap-ul lor folosindu-ne de registrii AX,BX
67              mov [sir_high+EDI],AX
68
69              mic:
70
71              add EDI,2                   ;din moment ce am ales sa parcugem manual sirul, va trebui sa incrementam EDI cu
72              dec byte[cont]              ;decrementam contorul
73              cmp byte[cont],0            ;implementam conditia de oprire a loop-ului
74          jge slp
75          add ESI,2                       ;incrementam manual si valoarea lui ESI
76      loop bg_loop
77
78      ;reconstituim sirul
```

```
bits 32 ; assembling for the 32 bits architecture
global start
extern exit               ; tell nasm that exit exists even if we won't be defining it
import exit msvcrt.dll    ; exit is a function that ends the calling process. It is
defined in msvcrt.dll
segment data use32 class=data
    sir DD 12AB5678h, 1256ABCDh, 12344344h ;definim sirul din problema
    l equ ($-sir)/4 ;lungimea sirului sir
    sir_high times l dw 0 ; acest sir va contine partea high a dw din sir
    sir_low times l dw 0 ;  acest sir va contine partea low a dw din sir
    rez times l dd 0
    cont db 0
segment code use32 class=code
    start:
        cld ;sirul va fi parcurs de la stanga la dreapta
        mov ECX,l ; punem in registrul contor, lungimea sirului
        mov ESI,sir ;punem in ESI adresa de inceput a sirului
        mov EDX,0   ;EDX va fi registru index pentru sir_high si sir_low
        jecxz sfarsit
        st_loop:
            lodsd  ;aduncem in EAX valoarea de la [sir:ESI]
            mov BX,AX ;aducem in BX partea low a dublucuvantului
           and EAX,0FFFF0000h ;izolam cuvantul high din EAX
            shr EAX,16 ;aducem dublucuvantul high in locul dublucuvantului low
            ;acum avem BX si in AX cele doua componente ale numarului cu care vom opera
            mov [sir_high+EDX],AX  ;punem partea high, stocata in AX in sir_high
            mov [sir_low+EDX] ,BX  ;punem partea low, stocata in BX in sir_low
            add EDX,2              ;incrementam cu 2 indexul sirurilor deoarece lucram cu
        loop st_loop
        sfarsit:
        ;vom proiecta un algoritm de sortare pentru sir_high
        mov ECX,l ;punem in registrul contor, lungimea sirului
        mov ESI,0 ;punem in ESI 0, pentru a parcurge sirul cu el
        bg_loop:
            mov EDI,0          ;initializam EDI cu 0, folosindul ca index pentru a realiza
            mov byte[cont],CL  ; folosim cont pe post de variabila contor, punem in ea nr
            dec byte[cont]     ; decrementam cu o unitate aferenta cu inceperea de la j+1
            stra:
            add EDI,2          ;il aducem prin aceasta structura pe EDI la valoarea lui SI
            cmp EDI,ESI
            jbe stra
            slp:
                mov AX,[sir_high+ESI]       ;preluam in AX elementul din sirul high
                cmp AX,word[sir_high+EDI]   ;comparam cu elementul de pe pozitia j
                jb mic
                mov BX,[sir_high+EDI]       ;in cazul in care am gasit doua elemente ce
                mov [sir_high+ESI],BX       ;realizam swap-ul lor folosindu-ne de
                mov [sir_high+EDI],AX
                mic:
                add EDI,2                   ;din moment ce am ales sa parcugem manual
                dec byte[cont]              ;decrementam contorul
                cmp byte[cont],0            ;implementam conditia de oprire a loop-ului
            jge slp
            add ESI,2                       ;incrementam manual si valoarea lui ESI
        loop bg_loop
```

```
;reconstituim sirul
        mov EDI,rez ; punem in EDI, adresa rezultatului, astfel incat sa il
construim automat
        mov ECX,l  ;punem in ECX lungimea sirului, pentru a realiza loop ul
        mov ESI,0  ;punem in ESI,0 deoarece vom parcuge manual acest sir

        lp_st:

            mov EAX,0
            mov AX,[sir_high+ESI]  ;aducem in AX(word ul low al EAX) partea high a
sirului

            shl EAX,16             ;shiftam spre stanga cu 16 pozitii in asa fel
incat ducem partea high pe pozitie
            mov AX,[sir_low+ESI]   ;adaugam partea low

            stosd                  ;stocam partea low in [rez:EDI]

            add ESI,2              ;incrementam manual indexul

        loop lp_st

        push    dword 0      ; push the parameter for exit onto the stack
        call    [exit]       ; call exit to terminate the program
```

DF=1

```
; our code starts here
segment code use32 class=code
    start:
        std ;sirul va fi parcurs de la stanga la dreapta
        mov ECX,l ; punem in registrul contor, lungimea sirului
        mov ESI,sir ;punem in ESI adresa de inceput a sirului
        add ESI,4*l-4  ;punem in ESI adresa finala a sirului prin adunarea lungimii
        mov EDX,0    ;EDX va fi registru index pentru sir_high si sir_low
        jecxz sfarsit

        st_loop:
            lodsd  ;aduncem in EAX valoarea de la [sir:ESI]
            mov BX,AX ;aducem in BX partea low a dublucuvantului

            and EAX,0FFFF0000h ;izolam cuvantul high din EAX
            shr EAX,16 ;aducem dublucuvantul high in locul dublucuvantului low
            ;acum avem BX si in AX cele doua componente ale numarului cu care vom opera mai departe

            mov [sir_high+EDX],AX  ;punem partea high, stocata in AX in sir_high
            mov [sir_low+EDX] ,BX  ;punem partea low, stocata in BX in sir_low
            add EDX,2              ;incrementam cu 2 indexul sirurilor deoarece lucram cu word-uri

        loop st_loop
        sfarsit:
        ;vom proiecta un algoritm de sortare pentru sir_high
        mov ECX,l ;punem in registrul contor, lungimea sirului
        mov ESI,0 ;punem in ESI 0, pentru a parcurge sirul cu el

        bg_loop:

            mov EDI,0          ;initializam EDI cu 0, folosindul ca index pentru a realiza sortare
            mov byte[cont],CL  ; folosim cont pe post de variabila contor, punem in ea nr ramas de pasi
            dec byte[cont]     ; decrementam cu o unitate aferenta cu inceperea de la j+1 a for-ului din sortarea prin selectie

            stra:
            add EDI,2          ;il aducem prin aceasta structura pe EDI la valoarea lui ESI
            cmp EDI,ESI
            jbe stra

            slp:

                mov AX,[sir_high+ESI]       ;preluam in AX elementul din sirul high (elementul de pe pozitia i)
                cmp AX,word[sir_high+EDI]   ;comparam cu elementul de pe pozitia j

                jb mic

                mov BX,[sir_high+EDI]       ;in cazul in care am gasit doua elemente ce trebuie interschimbate
                mov [sir_high+ESI],BX       ;realizam swap-ul lor folosindu-ne de registrii AX,BX
```



```
bits 32 ; assembling for the 32 bits architecture

; declare the EntryPoint (a label defining the very first instruction of the program)
global start

; declare external functions needed by our program
extern exit               ; tell nasm that exit exists even if we won't be defining it
import exit msvcrt.dll    ; exit is a function that ends the calling process. It is
defined in msvcrt.dll
                          ; msvcrt.dll contains exit, printf and all the other important
C-runtime specific functions

; our data is declared here (the variables needed by our program)
segment data use32 class=data
    sir DD 12AB5678h, 1256ABCDh, 12344344h ;definim sirul din problema
    l equ ($-sir)/4 ;lungimea sirului sir
    sir_high times l dw 0 ; acest sir va contine partea high a dw din sir
    sir_low times l dw 0 ;  acest sir va contine partea low a dw din sir
    rez times l dd 0
    cont db 0

; our code starts here
segment code use32 class=code
    start:
        std ;sirul va fi parcurs de la stanga la dreapta
        mov ECX,l ; punem in registrul contor, lungimea sirului
        mov ESI,sir ;punem in ESI adresa de inceput a sirului
        add ESI,4*l-4  ;punem in ESI adresa finala a sirului prin adunarea lungimii
        mov EDX,0   ;EDX va fi registru index pentru sir_high si sir_low
        jecxz sfarsit

        st_loop:
            lodsd  ;aduncem in EAX valoarea de la [sir:ESI]
            mov BX,AX ;aducem in BX partea low a dublucuvantului

            and EAX,0FFFF0000h ;izolam cuvantul high din EAX
            shr EAX,16 ;aducem dublucuvantul high in locul dublucuvantului low
            ;acum avem BX si in AX cele doua componente ale numarului cu care vom opera
mai departe

            mov [sir_high+EDX],AX  ;punem partea high, stocata in AX in sir_high
            mov [sir_low+EDX] ,BX  ;punem partea low, stocata in BX in sir_low
            add EDX,2              ;incrementam cu 2 indexul sirurilor deoarece lucram cu
word-uri

        loop st_loop
        sfarsit:
        ;vom proiecta un algoritm de sortare pentru sir_high
        mov ECX,l ;punem in registrul contor, lungimea sirului
        mov ESI,0 ;punem in ESI 0, pentru a parcurge sirul cu el
```

```
bg_loop:
            mov EDI,0          ;initializam EDI cu 0, folosindul ca index pentru a realiza
sortare
            mov byte[cont],CL  ; folosim cont pe post de variabila contor, punem in ea nr
ramas de pasi
            dec byte[cont]     ; decrementam cu o unitate aferenta cu inceperea de la j+1
a for-ului din sortarea prin selectie
            stra:
            add EDI,2          ;il aducem prin aceasta structura pe EDI la valoarea lui
ESI
            cmp EDI,ESI
            jbe stra
            slp:
                mov AX,[sir_high+ESI]       ;preluam in AX elementul din sirul high
(elementul de pe pozitia i)
                cmp AX,word[sir_high+EDI]   ;comparam cu elementul de pe pozitia j
                jb mic
                mov BX,[sir_high+EDI]       ;in cazul in care am gasit doua elemente ce
trebuie interschimbate
                mov [sir_high+ESI],BX       ;realizam swap-ul lor folosindu-ne de
registrii AX,BX
                mov [sir_high+EDI],AX

                mic:

                add EDI,2                   ;din moment ce am ales sa parcugem manual
sirul, va trebui sa incrementam EDI cu
                dec byte[cont]              ;decrementam contorul
                cmp byte[cont],0            ;implementam conditia de oprire a loop-ului
            jge slp
            add ESI,2                       ;incrementam manual si valoarea lui ESI
        loop bg_loop
        ;reconstituim sirul
        mov EDI,rez ;punem in EDI, adresa rezultatului, astfel incat sa il construim
automat
        add EDI,4*l-4 ;intrucat parcurgem de la capat sirul, vom pune ESI adresa finala a
acestuia
        mov ECX,l  ;punem in ECX lungimea sirului, pentru a realiza loop ul
        mov ESI,0  ;punem in ESI,0 deoarece vom parcuge manual acest sir
        lp_st:

            mov EAX,0
            mov AX,[sir_high+ESI]  ;aducem in AX(word ul low al EAX) partea high a sirului

            shl EAX,16             ;shiftam spre stanga cu 16 pozitii in asa fel incat
ducem partea high pe pozitie
            mov AX,[sir_low+ESI]   ;adaugam partea low

            stosd                  ;stocam partea low in [rez:EDI]

            add ESI,2              ;incrementam manual indexul
        loop lp_st

        push    dword 0      ; push the parameter for exit onto the stack
        call    [exit]       ; call exit to terminate the program
```