

```

32      mov BX,AX ;aducem in BX partea low a dublucuvantului
33
34      and EAX,0FFFF0000h ;izolam cuvantul high din EAX
35      shr EAX,16 ;aducem dublucuvantul high in locul dublucuvantului low
36      ;acum avem BX si in AX cele doua componente ale numarului cu care vom opera mai departe
37
38      mov [sir_high+EDI],AX ;punem partea high, stocata in AX in sir_high
39      mov [sir_low+EDI],BX ;punem partea low, stocata in BX in sir_low
40      add EDI,2 ;incrementam cu 2 indexul sirurilor deoarece lucram cu word-uri
41
42 loop st_loop
43 sfarsit:
44 ;vom proiecta un algoritim de sortare pentru sir_high
45 mov ECX,1 ;punem in registrul contor, lungimea sirului
46 mov ESI,0 ;punem in ESI 0, pentru a parcurge sirul cu el
47
48 bg_loop:
49
50      mov EDI,0 ;initializam EDI cu 0, folosindu-l ca index pentru a realiza sortare
51      mov byte[cont],CL ; folosim cont pe post de variabila contor, punem in ea nr ramas de pasi
52      dec byte[cont] ;decrementam cu o unitate aferenta cu inceperea de la j+1 a for-ului din sortarea prin selectie
53
54      stra:
55      add EDI,2 ;il aducem prin aceasta structura pe EDI la valoarea lui ESI
56      cmp EDI,ESI
57      jbe stra
58
59      slp:
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 registrul AX,BX
67      mov [sir_high+EDI],AX
68
69      mic:
70
71      add EDI,2 ;din moment ce am ales sa parcurgem 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 1 dw 0 ; acest sir va contine partea high a dw din sir
    sir_low times 1 dw 0 ; acest sir va contine partea low a dw din sir
    rez times 1 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,1 ;punem in ECX lungimea sirului, pentru a realiza loop ul
    mov ESI,0 ;punem in ESI,0 deoarece vom parcurge 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

```
20 ; our code starts here
21 segment code use32 class=code
22 start:
23     std ;sirul va fi parcurs de la stanga la dreapta
24     mov ECX,1 ; punem in registrul contor, lungimea sirului
25     mov ESI,sir ;punem in ESI adresa de inceput a sirului
26     add ESI,*1-4 ;punem in ESI adresa finala a sirului prin adunarea lungimii
27     mov EDI,0 ;EDI va fi registru index pentru sir_high si sir_low
28     jecxz sfarsit
29
30     st_loop:
31         lodsd ;aducem in EAX valoarea de la [sir:ESI]
32         mov BX,AX ;aducem in BX partea low a dublucuvantului
33
34         and EAX,0FFFF0000h ;izolam cuvantul high din EAX
35         shr EAX,16 ;aducem dublucuvantul high in locul dublucuvantului low
36         ;acum avem BX si in AX cele doua componente ale numarului cu care vom opera mai departe
37
38         mov [sir_high+EDI],AX ;punem partea high, stocata in AX in sir_high
39         mov [sir_low+EDI],BX ;punem partea low, stocata in BX in sir_low
40         add EDI,2 ;incrementam cu 2 indexul sirurilor deoarece lucram cu word-uri
41
42     loop st_loop
43     sfarsit:
44     ;vom proiecta un algoritim de sortare pentru sir_high
45     mov ECX,1 ;punem in registrul contor, lungimea sirului
46     mov ESI,0 ;punem in ESI 0, pentru a parcurge sirul cu el
47
48     bg_loop:
49
50         mov EDI,0 ;initializam EDI cu 0, folosindu-l ca index pentru a realiza sortare
51         mov byte[cont],CL ; folosim cont pe post de variabila contor, punem in ea nr ramas de pasi
52         dec byte[cont] ; decrementam cu o unitate aferenta cu inceperea de la j+1 a for-ului din sortarea prin selectie
53
54     stra:
55         add EDI,2 ;il aducem prin aceasta structura pe EDI la valoarea lui ESI
56         cmp EDI,ESI
57         jbe stra
58
59     slp:
60
61         mov AX,[sir_high+ESI] ;preluam in AX elementul din sirul high (elementul de pe pozitia i)
62         cmp AX,word[sir_high+EDI] ;comparam cu elementul de pe pozitia j
63
64         jb mic
65
66         mov BX,[sir_high+EDI] ;in cazul in care am gasit doua elemente ce trebuie interschimbate
67         mov [sir_high+ESI],BX ;realizam swap-ul lor folosindu-ne de registrii AX,BX
```

Assembly language source file length: 4,922 lines: 101 Ln:26 Col:58 Sel:0|0 Windows (CR LF) UTF-8 INS

00401000	00000000	MOV	ECX	,	1
00401001	00000000	MOV	ESI	,	OFFSET 00401000
00401002	00000000	MOV	ECX	,	1
00401003	00000000	MOV	ESI	,	0
00401004	00000000	MOV	ESI	,	0
00401005	00000000	MOV	ESI	,	0
00401006	00000000	MOV	ESI	,	0
00401007	00000000	MOV	ESI	,	0
00401008	00000000	MOV	ESI	,	0
00401009	00000000	MOV	ESI	,	0
0040100A	00000000	MOV	ESI	,	0
0040100B	00000000	MOV	ESI	,	0
0040100C	00000000	MOV	ESI	,	0
0040100D	00000000	MOV	ESI	,	0
0040100E	00000000	MOV	ESI	,	0
0040100F	00000000	MOV	ESI	,	0
00401010	00000000	MOV	ESI	,	0
00401011	00000000	MOV	ESI	,	0
00401012	00000000	MOV	ESI	,	0
00401013	00000000	MOV	ESI	,	0
00401014	00000000	MOV	ESI	,	0
00401015	00000000	MOV	ESI	,	0
00401016	00000000	MOV	ESI	,	0
00401017	00000000	MOV	ESI	,	0
00401018	00000000	MOV	ESI	,	0
00401019	00000000	MOV	ESI	,	0
0040101A	00000000	MOV	ESI	,	0
0040101B	00000000	MOV	ESI	,	0
0040101C	00000000	MOV	ESI	,	0
0040101D	00000000	MOV	ESI	,	0
0040101E	00000000	MOV	ESI	,	0
0040101F	00000000	MOV	ESI	,	0
00401020	00000000	MOV	ESI	,	0
00401021	00000000	MOV	ESI	,	0
00401022	00000000	MOV	ESI	,	0
00401023	00000000	MOV	ESI	,	0
00401024	00000000	MOV	ESI	,	0
00401025	00000000	MOV	ESI	,	0
00401026	00000000	MOV	ESI	,	0
00401027	00000000	MOV	ESI	,	0
00401028	00000000	MOV	ESI	,	0
00401029	00000000	MOV	ESI	,	0
0040102A	00000000	MOV	ESI	,	0
0040102B	00000000	MOV	ESI	,	0
0040102C	00000000	MOV	ESI	,	0
0040102D	00000000	MOV	ESI	,	0
0040102E	00000000	MOV	ESI	,	0
0040102F	00000000	MOV	ESI	,	0
00401030	00000000	MOV	ESI	,	0
00401031	00000000	MOV	ESI	,	0
00401032	00000000	MOV	ESI	,	0
00401033	00000000	MOV	ESI	,	0
00401034	00000000	MOV	ESI	,	0
00401035	00000000	MOV	ESI	,	0
00401036	00000000	MOV	ESI	,	0
00401037	00000000	MOV	ESI	,	0
00401038	00000000	MOV	ESI	,	0
00401039	00000000	MOV	ESI	,	0
0040103A	00000000	MOV	ESI	,	0
0040103B	00000000	MOV	ESI	,	0
0040103C	00000000	MOV	ESI	,	0
0040103D	00000000	MOV	ESI	,	0
0040103E	00000000	MOV	ESI	,	0
0040103F	00000000	MOV	ESI	,	0
00401040	00000000	MOV	ESI	,	0
00401041	00000000	MOV	ESI	,	0
00401042	00000000	MOV	ESI	,	0
00401043	00000000	MOV	ESI	,	0
00401044	00000000	MOV	ESI	,	0
00401045	00000000	MOV	ESI	,	0
00401046	00000000	MOV	ESI	,	0
00401047	00000000	MOV	ESI	,	0
00401048	00000000	MOV	ESI	,	0
00401049	00000000	MOV	ESI	,	0
0040104A	00000000	MOV	ESI	,	0
0040104B	00000000	MOV	ESI	,	0
0040104C	00000000	MOV	ESI	,	0
0040104D	00000000	MOV	ESI	,	0
0040104E	00000000	MOV	ESI	,	0
0040104F	00000000	MOV	ESI	,	0
00401050	00000000	MOV	ESI	,	0
00401051	00000000	MOV	ESI	,	0
00401052	00000000	MOV	ESI	,	0
00401053	00000000	MOV	ESI	,	0
00401054	00000000	MOV	ESI	,	0
00401055	00000000	MOV	ESI	,	0
00401056	00000000	MOV	ESI	,	0
00401057	00000000	MOV	ESI	,	0
00401058	00000000	MOV	ESI	,	0
00401059	00000000	MOV	ESI	,	0
0040105A	00000000	MOV	ESI	,	0
0040105B	00000000	MOV	ESI	,	0
0040105C	00000000	MOV	ESI	,	0
0040105D	00000000	MOV	ESI	,	0
0040105E	00000000	MOV	ESI	,	0
0040105F	00000000	MOV	ESI	,	0
00401060	00000000	MOV	ESI	,	0
00401061	00000000	MOV	ESI	,	0
00401062	00000000	MOV	ESI	,	0
00401063	00000000	MOV	ESI	,	0
00401064	00000000	MOV	ESI	,	0
00401065	00000000	MOV	ESI	,	0
00401066	00000000	MOV	ESI	,	0
00401067	00000000	MOV	ESI	,	0
00401068	00000000	MOV	ESI	,	0
00401069	00000000	MOV	ESI	,	0
0040106A	00000000	MOV	ESI	,	0
0040106B	00000000	MOV	ESI	,	0
0040106C	00000000	MOV	ESI	,	0
0040106D	00000000	MOV	ESI	,	0
0040106E	00000000	MOV	ESI	,	0
0040106F	00000000	MOV	ESI	,	0
00401070	00000000	MOV	ESI	,	0
00401071	00000000	MOV	ESI	,	0
00401072	00000000	MOV	ESI	,	0
00401073	00000000	MOV	ESI	,	0
00401074	00000000	MOV	ESI	,	0
00401075	00000000	MOV	ESI	,	0
00401076	00000000	MOV	ESI	,	0
00401077	00000000	MOV	ESI	,	0
00401078	00000000	MOV	ESI	,	0
00401079	00000000	MOV	ESI	,	0
0040107A	00000000	MOV	ESI	,	0
0040107B	00000000	MOV	ESI	,	0
0040107C	00000000	MOV	ESI	,	0
0040107D	00000000	MOV	ESI	,	0
0040107E	00000000	MOV	ESI	,	0
0040107F	00000000	MOV	ESI	,	0
00401080	00000000	MOV	ESI	,	0
00401081	00000000	MOV	ESI	,	0
00401082	00000000	MOV	ESI	,	0
00401083	00000000	MOV	ESI	,	0
00401084	00000000	MOV	ESI	,	0
00401085	00000000	MOV	ESI	,	0
00401086	00000000	MOV	ESI	,	0
00401087	00000000	MOV	ESI	,	0
00401088	00000000	MOV	ESI	,	0
00401089	00000000	MOV	ESI	,	0
0040108A	00000000	MOV	ESI	,	0
0040108B	00000000	MOV	ESI	,	0
0040108C	00000000	MOV	ESI	,	0
0040108D	00000000	MOV	ESI	,	0
0040108E	00000000	MOV	ESI	,	0
0040108F	00000000	MOV	ESI	,	0
00401090	00000000	MOV	ESI	,	0
00401091	00000000	MOV	ESI	,	0
00401092	00000000	MOV	ESI	,	0
00401093	00000000	MOV	ESI	,	0
00401094	00000000	MOV	ESI	,	0
00401095	00000000	MOV	ESI	,	0
00401096	00000000	MOV	ESI	,	0
00401097	00000000	MOV	ESI	,	0
00401098	00000000	MOV	ESI	,	0
00401099	00000000	MOV	ESI	,	0
0040109A	00000000	MOV	ESI	,	0
0040109B	00000000	MOV	ESI	,	0
0040109C	00000000	MOV	ESI	,	0
0040109D	00000000	MOV	ESI	,	0
0040109E	00000000	MOV	ESI	,	0
0040109F	00000000	MOV	ESI	,	0
004010A0	00000000	MOV	ESI	,	0
004010A1	00000000	MOV	ESI	,	0
004010A2	00000000	MOV	ESI	,	0
004010A3	00000000	MOV	ESI	,	0
004010A4	00000000	MOV	ESI	,	0
004010A5	00000000	MOV	ESI	,	0
004010A6	00000000	MOV	ESI	,	0
004010A7	00000000	MOV	ESI	,	0
004010A8	00000000	MOV	ESI	,	0
004010A9	00000000	MOV	ESI	,	0
004010AA	00000000	MOV	ESI	,	0
004010AB	00000000	MOV	ESI	,	0
004010AC	00000000	MOV	ESI	,	0
004010AD	00000000	MOV	ESI	,	0
004010AE	00000000	MOV	ESI	,	0
004010AF	00000000	MOV	ESI	,	0
004010B0	00000000	MOV	ESI	,	0
004010B1	00000000	MOV	ESI	,	0
004010B2	00000000	MOV	ESI	,	0
004010B3	00000000	MOV	ESI	,	0
004010B4	00000000	MOV	ESI	,	0
004010B5	00000000	MOV	ESI	,	0
004010B6	00000000	MOV	ESI	,	0
004010B7	00000000	MOV	ESI	,	0
004010B8	00000000	MOV	ESI	,	0
004010B9	00000000	MOV	ESI	,	0
004010BA	00000000	MOV	ESI	,	0
004010BB	00000000	MOV	ESI	,	0
004010BC	00000000	MOV	ESI	,	0
004010BD	00000000	MOV	ESI	,	0
004010BE	00000000	MOV	ESI	,	0
004010BF	00000000	MOV	ESI	,	0
004010C0	00000000	MOV	ESI	,	0
004010C1	00000000	MOV	ESI	,	0
004010C2	00000000	MOV	ESI	,	0
004010C3	00000000	MOV	ESI	,	0
004010C4	00000000	MOV	ESI	,	0
004010C5	00000000	MOV	ESI	,	0
004010C6	00000000	MOV	ESI	,	0
004010C7	00000000	MOV	ESI	,	0
004010C8	00000000	MOV	ESI	,	0
004010C9	00000000	MOV	ESI	,	0
004010CA	00000000	MOV	ESI	,	0
004010CB	00000000	MOV	ESI	,	0
004010CC	00000000	MOV	ESI	,	0
004010CD	00000000	MOV	ESI	,	0
004010CE	00000000	MOV	ESI	,	0
004010CF	00000000	MOV	ESI	,	0
004010D0	00000000	MOV	ESI	,	0
004010D1	00000000	MOV	ESI	,	0
004010D2	00000000	MOV	ESI	,	0
004010D3	00000000	MOV	ESI	,	0
004010D4	00000000	MOV	ESI	,	0
004010D5	00000000	MOV	ESI	,	0
004010D6	00000000	MOV	ESI	,	0
004010D7	00000000	MOV	ESI	,	0
004010D8	00000000	MOV	ESI	,	0
004010D9	00000000	MOV	ESI	,	0
004010DA	00000000	MOV	ESI	,	0
004010DB	00000000	MOV	ESI	,	0
004010DC	00000000	MOV	ESI	,	0
004010DD	00000000	MOV	ESI	,	0
004010DE	00000000	MOV	ESI	,	0
004010DF	00000000	MOV	ESI	,	0
004010E0	00000000	MOV	ESI	,	0
004010E1	00000000	MOV	ESI	,	0
004010E2	00000000	MOV	ESI	,	0
004010E3	00000000	MOV	ESI	,	0
004010E4	00000000	MOV	ESI	,	0
004010E5	00000000	MOV	ESI	,	0
004010E6	00000000	MOV	ESI	,	0
004010E7	00000000	MOV	ESI	,	0
004010E8	00000000	MOV	ESI	,	0
004010E9	00000000	MOV	ESI	,	0
004010EA	00000000	MOV	ESI	,	0
004010EB	00000000	MOV	ESI	,	0
004010EC	00000000	MOV	ESI	,	0
004010ED	00000000	MOV	ESI	,	0
004010EE	0000000				

```

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 1 dw 0 ; acest sir va contine partea high a dw din sir
    sir_low times 1 dw 0 ; acest sir va contine partea low a dw din sir
    rez times 1 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*1-4 ;intrucat parcurgem de la capat sirul, vom pune ESI adresa finala a
acestuia
mov ECX,1 ;punem in ECX lungimea sirului, pentru a realiza loop ul
mov ESI,0 ;punem in ESI,0 deoarece vom parcuze 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

```