A picture containing text

Description automatically generated

Diagram

Description automatically generated

Text

Description automatically generated

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

Text

Description automatically generatedA picture containing diagram

Description automatically generated

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