;18. Se dau 2 siruri A si B. Sa se construiasca sirul R care sa contina doar elementele impare si pozitive din cele 2 siuri

;Exemplu:

;A:2,1,3,-3

;B:4,5,-5,7

;R:1,3,5,7

bits 32

global start

extern exit

import exit msvcrt.dll

segment data use32 class=data

a db -2,1,3,-3

lga equ $-a

b db 4,5,-6,7

lgb equ $-b

lgr equ $-a

r times lgr db 0

segment code use32 class=code

start:

mov ecx, lga ;pun lungimea in ecx pentru a putea realiza bucla loop de ecx ori

jecxz parcurgereb

mov esi, 0

mov edi, 0

repeta1:

mov al, [a+esi]

add al, 0

js finalepeta1 ;testez daca elementul e negativ

test al, 00000001b ;testez daca elementul e impar

jnz cazimpar1

jmp finalepeta1

cazimpar1:

mov [r+edi], al

inc edi

finalepeta1:

inc esi

loop repeta1

parcurgereb:

mov ecx, lgb

jecxz final

mov esi, 0

repeta2:

mov al, [b+esi]

add al, 0

js finalepeta2

test al, 00000001b

jnz cazimpar2

jmp finalepeta2

cazimpar2:

mov [r+edi], al

inc edi

finalepeta2:

inc esi

loop repeta2

final:

push dword 0

call [exit]

**Temă în plus**

;32. Se da un sir de octeti S de lungime l. Sa se construiasca sirul D de ;lungime l-1 astfel incat elementele din D sa reprezinte catul dintre fiecare 2 ;elemente consecutive S(i) si S(i+1) din S.

;Exemplu:

;S: 1, 6, 3, 1

;D: 0, 2, 3

bits 32

global start

extern exit

import exit msvcrt.dll

segment data use32 class=data

s db 1,6,3,1

lg equ $-s

d times lg db 0

segment code use32 class=code

start:

mov ecx, lg

sub ecx, 1

jecxz final

mov edi, 0

mov esi, 1

repeta:

mov al, [s+edi]

cbw

mov bl, [s+esi]

idiv bl

mov [d+edi], ax

inc esi

inc edi

loop repeta

final:

push dword 0

call [exit]

;20. Se dau 2 siruri de octeti A si B. Sa se construiasca sirul R care sa contina ;elementele lui B in ordine inversa urmate de elementele pare ale lui A.

;Exemplu:

;A: 2, 1, 3, 3, 4, 2, 6

;B: 4, 5, 7, 6, 2, 1

;R: 1, 2, 6, 7, 5, 4, 2, 4, 2, 6

bits 32

global start

extern exit

import exit msvcrt.dll

segment data use32 class=data

a db 2,1,3,3,4,2,6

lga equ $-a

b db 4,5,7,6,2,1

lgb equ $-b

lg equ $-a

r times lg db 0

segment code use32 class=code

start:

mov ecx, lgb

jecxz parcurgerea

mov esi, lgb

sub esi, 1

mov edi, 0

repeta1:

mov al, [b+esi]

mov [r+edi], al

dec esi

inc edi

loop repeta1

parcurgerea:

mov ecx, lga

jecxz final

mov esi, 0

repeta2:

mov al, [a+esi]

test al, 000000001b

jz cazpar

jmp finalrepeta2

cazpar:

mov al, [a+esi]

mov [r+edi], al

inc edi

finalrepeta2:

inc esi

loop repeta2

final:

push dword 0

call [exit]

;27. Se dau 2 siruri de octeti S1 si S2 de aceeasi lungime. Sa se construiasca sirul D astfel incat fiecare element din D sa reprezinte diferenta dintre elementele de pe pozitiile corespunzatoare din S1 si S2.

;Exemplu:

;S1: 1, 3, 6, 2, 3, 2

;S2: 6, 3, 8, 1, 2, 5

;D: -5, 0, -2, 1, 1, -3

bits 32

global start

extern exit

import exit msvcrt.dll

segment data use32 class=data

s1 db 1,3,6,2,3,2

lgs1 equ $-s1

s2 db 6,3,8,1,2,5

lg equ $-s1

d times lg db 0

segment code use32 class=code

start:

mov ecx, lgs1

jecxz final

mov esi, 0

mov edi, 0

repeta:

mov al, [s1+esi]

sub al, [s2+esi]

mov [d+edi], al

inc edi

inc esi

loop repeta

final:

push dword 0

call [exit]

;17. Se dau 2 siruri de octeti S1 si S2 de aceeasi lungime. Sa se construiasca sirul D astfel incat fiecare element din D sa reprezinte maximul dintre elementele de pe pozitiile corespunzatoare din S1 si S2.

;S1: 1, 3, 6, 2, 3, 7

;S2: 6, 3, 8, 1, 2, 5

;D: 6, 3, 8, 2, 3, 7

bits 32

global start

extern exit

import exit msvcrt.dll

segment data use32 class=data

s1 db 1,3,6,2,3,7

lgs1 equ $-s1

s2 db 6,3,8,1,2,5

lg equ $-s1

d times lg db 0

segment code use32 class=code

start:

mov ecx, lgs1

jecxz final

mov esi, 0

mov edi, 0

repeta:

mov al, [s1+esi]

mov bl, [s2+esi]

cmp al, bl

ja adaugas1

mov [d+edi], bl

jmp finaladauga

adaugas1:

mov [d+edi], al

jmp finaladauga

finaladauga:

inc esi

inc edi

loop repeta

final:

push dword 0

call [exit]