

UNIVERSITATEA DE STAT DIN MOLDOVA

**FACULTATEA MATEMATICĂ ȘI INFORMATICĂ
DEPARTAMENTUL INFORMATICĂ**

CEMÎRTAN CRISTIAN

Lucrarea individuală nr. 5
la disciplina *Arhitectura Calculatoarelor și Limbaje de Asamblare*

Coordonator: Sturza Greta, lector universitar

Chișinău, 2021

Cuprins

Sarcină.....	3
Cod sursă (versiunea .EXE).....	3
Cod sursă (versiunea .COM).....	5
Rezultate.....	7
Depanare.....	8
Concluzie.....	12

Sarcină

Este definit șirul **s1**. Să se scrie un program ce introduce de la tastatură simbolul **a1**, șirurile **s2**, **s3** și afișează concatenarea elementelor introduse în modul următor:

```
Introduceti sirul s2:
s2
Introduceti simbolul a1: a1
Introduceti sirul s3: s3
Rezultatul obtinut: s1bbals1a1bbs2s3
```

Cod sursă (versiunea .EXE)

```
COMMENT *
    Lucrare individuala nr. 5, varianta 3
    Versiunea .EXE
    Copyright Cemirtan Cristian 2021
    Grupa I 2101
*

.MODEL small
.STACK

len      EQU 10
crlf     EQU 0Dh, 0Ah

.DATA
DB crlf
txt_sir DB 'Introduceti sirul s'
cifra DB '2', \ ; truc pentru a nu crea mesaje redundante
':', crlf, '$'

txt_sim DB 0Ah, 'Introduceti simbolul a1: $'
txt_rez DB 0Ah, 'Rezultatul obtinut: $'

b DB '$' ; spatiu liber
a1 DB ?, '$' ; caracter
s1 DB 'definit$' ; sirul s1 este definit

msk DW \
    OFFSET s3, OFFSET s2, OFFSET b, OFFSET b, OFFSET a1, \
    OFFSET s1, OFFSET a1, OFFSET b, OFFSET b, OFFSET s1, \
    OFFSET txt_rez

; tabelul de cautare
; se interpreteaza de la dreapta la stanga

msk_idx EQU $ - msk - 2 ; pozitia ultimului pointer
```

```

lungime_s2_max DB len + 1
lungime_s2 DB ?
s2 DB len + 2 DUP (?)

lungime_s3_max DB ?
lungime_s3 DB ?
s3 DB len + 2 DUP (?)

.CODE
; initializarea segmentului de date
    mov dx, @data
    mov ds, dx

; optimizari de spatiu
    lea si, lungime_s2_max
    lea di, lungime_s3_max
    mov BYTE PTR [di], len + 1
    xor bx, bx ; liniile 66 si 92

; afisare text pentru sir #2
    mov ah, 09h
    lea dx, txt_sir
    int 21h

; citeste s2
    inc ah ; acum e 0Ah
    mov dx, si
    int 21h

; inlocuim in s2 0Dh cu '$'
    mov bl, [si + 1]
    mov BYTE PTR [si + bx + 2], '$'

; afisare text pentru caracter
    dec ah ; acum e 09h
    lea dx, txt_sim
    int 21h

; citeste a1
    mov ah, 1
    int 21h
    mov [a1], al

; formatam text pentru sir #3, conform cerintele sarcinii
    inc cifra
    mov WORD PTR [cifra + 2], '$ '

; afisare text pentru sir #3
    mov ah, 09h
    lea dx, [txt_sir - 2]
    int 21h

; citeste s3
    inc ah
    mov dx, di
    int 21h

```

```

; inlocuim in s3 0Dh cu '$'
    mov bl, lungime_s3
    mov BYTE PTR [di + bx + 2], '$'

; afisare rezultat
    dec ah
    lea si, msk
    mov bx, msk_idx

bucla:
    mov dx, [bx + si]
    int 21h
    sub bx, 2
    jnc bucla

; iesire cu succes, cu cod 0
    mov ax, 4C00h
    int 21h
END

```

Cod sursă (versiunea .COM)

```

COMMENT *
    Lucrare individuala nr. 5, varianta 3
    Versiunea .COM
    Copyright Cemirtan Cristian 2021
    Grupa I 2101
*

.MODEL tiny

.CODE
    ORG 100h

main:
; optimizari de spatiu
    lea si, lungime_s2_max
    lea di, lungime_s3_max
    mov BYTE PTR [di], len + 1
    xor bx, bx ; liniile 66 si 92

; afisare text pentru sir #2
    mov ah, 09h
    lea dx, txt_sir
    int 21h

; citeste s2
    inc ah ; acum e 0Ah
    mov dx, si
    int 21h

; inlocuim in s2 0Dh cu '$'
    mov bl, [si + 1]

```

```

        mov BYTE PTR [si + bx + 2], '$'

; afisare text pentru caracter
        dec ah ; acum e 09h
        lea dx, txt_sim
        int 21h

; citeste a1
        mov ah, 1
        int 21h
        mov [a1], al

; formatam text pentru sir #3, conform cerintele sarcinii
        inc cifra
        mov WORD PTR [cifra + 2], '$ '

; afisare text pentru sir #3
        mov ah, 09h
        lea dx, [txt_sir - 2]
        int 21h

; citeste s3
        inc ah
        mov dx, di
        int 21h

; inlocuim in s3 0Dh cu '$'
        mov bl, lungime_s3
        mov BYTE PTR [di + bx + 2], '$'

; afisare rezultat
        dec ah
        lea si, msk
        mov bx, msk_idx

buccla:
        mov dx, [bx + si]
        int 21h
        sub bx, 2
        jnc buccla

; iesire cu succes, cu cod 0
        mov ax, 4C00h
        int 21h

; date

len      EQU 10
crlf     EQU 0Dh, 0Ah

DB crlf
txt_sir DB 'Introduceti sirul s'
cifra DB '2', \ ; truc pentru a nu crea mesaje redundante
':', crlf, '$'

txt_sim DB 0Ah, 'Introduceti simbolul a1: $'

```

```

txt_rez DB 0Ah, 'Rezultatul obtinut: $'

b DB ' $' ; spatiu liber
a1 DB ?, '$' ; caracter
s1 DB 'definit$' ; sirul s1 este definit

msk DW \
        OFFSET s3, OFFSET s2, OFFSET b, OFFSET b, OFFSET a1, \
        OFFSET s1, OFFSET a1, OFFSET b, OFFSET b, OFFSET s1, \
        OFFSET txt_rez

; tabelul de cautare
; se interpreteaza de la dreapta la stanga

msk_idx EQU $ - msk - 2 ; pozitia ultimului pointer

lungime_s2_max DB len + 1
lungime_s2 DB ?
s2 DB len + 2 DUP (?)

lungime_s3_max DB ?
lungime_s3 DB ?
s3 DB len + 2 DUP (?)
END main

```

Rezultate

```

E:\>tasm i5
Turbo Assembler Version 3.0 Copyright (c) 1988, 1991 Borland International

Assembling file:    i5.ASM
Error messages:     None
Warning messages:   None
Passes:             1
Remaining memory:   451k

E:\>tlink i5
Turbo Link Version 2.0 Copyright (c) 1987, 1988 Borland International

E:\>i5
Introduceti sirul s2:
1234
Introduceti simbolul a1: "
Introduceti sirul s3: 5678
Rezultatul obtinut: definit "definit" 12345678
E:\>_

```

```

E:\>tasm i5com
Turbo Assembler Version 3.0 Copyright (c) 1988, 1991 Borland International

Assembling file:    i5com.ASM
Error messages:     None
Warning messages:   None
Passes:             1
Remaining memory:   451k

E:\>tlink /t i5com
Turbo Link Version 2.0 Copyright (c) 1987, 1988 Borland International

E:\>i5com.com
Introduceti sirul s2:
Versiunea
Introduceti simbolul a1: '
Introduceti sirul s3: .COM
Rezultatul obtinut: definit 'definit' Versiunea .COM
E:\>

```

Figurile 1 și 2. Programul executat cu succes.

Depanare

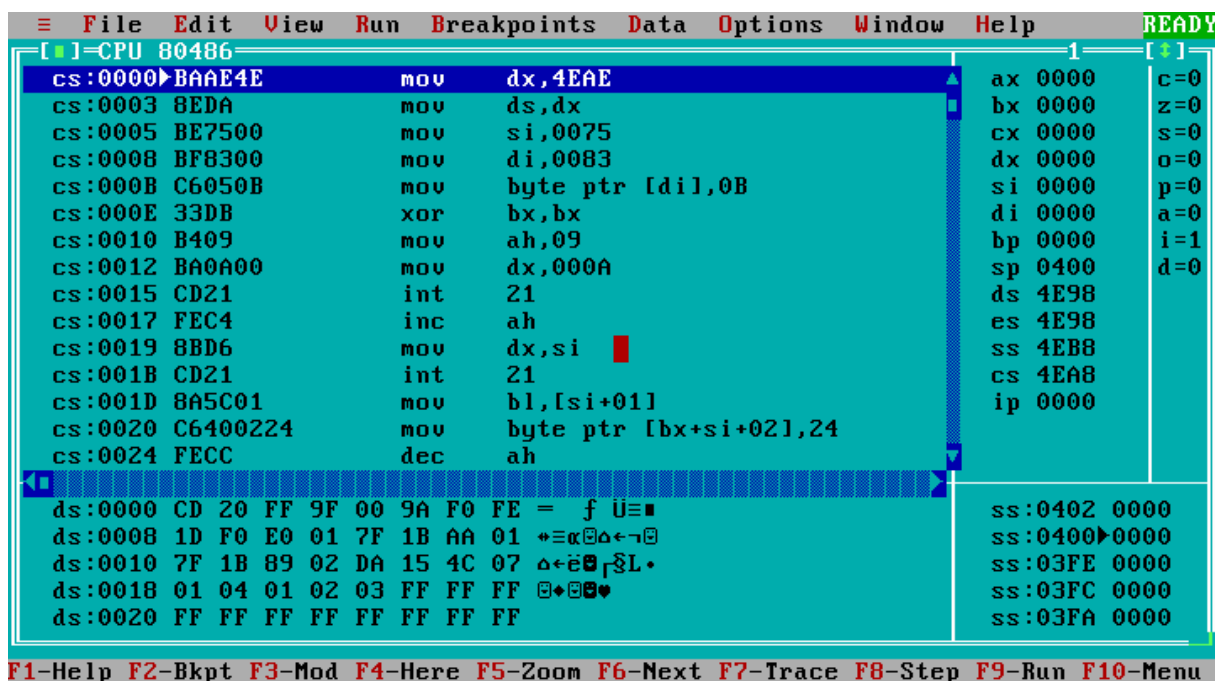


Figura 3. Depanatorul înainte de a rula programul sub format .EXE.

File Edit View Run Breakpoints Data Options Window Help

[CPU 80486]

Address	Instruction	Register	Value
cs:0100	mov	si	01D0
cs:0103	mov	di	01DE
cs:0106	mov	byte ptr [di],0B	
cs:0109	xor	bx,bx	
cs:010B	mov	ah,09	
cs:010D	mov	dx,0165	
cs:0110	int	21	
cs:0112	inc	ah	
cs:0114	mov	dx,si	
cs:0116	int	21	
cs:0118	mov	bl,[si+01]	
cs:011B	mov	byte ptr [bx+si+02],24	
cs:011F	dec	ah	
cs:0121	mov	dx,017D	
cs:0124	int	21	

Register	Value
ax	0000
bx	0000
cx	0000
dx	0000
si	0000
di	0000
bp	0000
sp	FFFE
ds	4E98
es	4E98
ss	4E98
cs	4E98
ip	0100

Address	Value
ds:0000	CD 20 FF 9F 00 9A F0 FE = f Ü=
ds:0008	1D F0 E0 01 7F 1B AA 01 *≡αΔ←→
ds:0010	7F 1B 89 02 DA 15 4C 07 Δ←ēΔ┘\$L•
ds:0018	01 04 01 02 03 FF FF FF Δ+ΔΔ♥
ds:0020	FF FF FF FF FF FF FF

ss:0000 20CD
ss:FFFE 0000
ss:FFFC 0000
ss:FFFA 0000
ss:FFF8 0000

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

Figura 4. Depanatorul înainte de a rula programul sub format .COM. În deosebire de versiunea .EXE, nu necesită încărcarea manuală a registrului ds, deoarece modelul *small* unifică stiva, cod și date într-un singur segment.

- În afară de prologul programului, versiunile .EXE și .COM sunt identice.
- Anterior, vom depana versiunea .EXE.

File Edit View Run Breakpoints Data Options Window Help

[CPU 80486]

Address	Instruction	Register	Value
cs:0000	mov	dx	4EAE
cs:0003	mov	ds,dx	
cs:0005	mov	si	0075
cs:0008	mov	di	0083
cs:000B	mov	byte ptr [di],0B	
cs:000E	xor	bx,bx	
cs:0010	mov	ah,09	
cs:0012	mov	dx,000A	
cs:0015	int	21	
cs:0017	inc	ah	
cs:0019	mov	dx,si	
cs:001B	int	21	
cs:001D	mov	bl,[si+01]	
cs:0020	mov	byte ptr [bx+si+02],24	
cs:0024	dec	ah	

Register	Value
ax	0000
bx	0000
cx	0000
dx	0000
si	0000
di	0000
bp	0000
sp	0400
ds	4E98
es	4E98
ss	4EB8
cs	4EA8
ip	0000

Address	Value
ds:0000	CD 20 FF 9F 00 9A F0 FE = f Ü=
ds:0008	1D F0 E0 01 7F 1B AA 01 *≡αΔ←→
ds:0010	7F 1B 89 02 DA 15 4C 07 Δ←ēΔ┘\$L•
ds:0018	01 04 01 02 03 FF FF FF Δ+ΔΔ♥
ds:0020	FF FF FF FF FF FF FF

ss:0402 0000
ss:0400 0000
ss:03FE 0000
ss:03FC 4EA8
ss:03FA 0018

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

Figura 5. Registrul ds modificat explicit, după executarea instrucțiunii aflată la adresa cs:0003h. Registrul ip indică locația instrucțiunii ce va fi executată.

```

E:\>td i5
Turbo Debugger Version 3.1 Copyright (c) 1988,92 Borland International
Introduceti sirul s2:
Individ 5_

```

Figura 6. Introducem de la tastatură, șirul s2. Întreruperea a avut loc la adresa cs:001Bh.

The screenshot shows the Turbo Debugger interface. The top menu bar includes File, Edit, View, Run, Breakpoints, Data, Options, Window, and Help. The status bar at the bottom shows function key shortcuts: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, and F10-Menu.

The main window displays assembly code for the CPU segment 80486. The code is as follows:

Address	Disassembly	Registers	Flags
cs:0010 B409	mov ah,09	ax 0A24	c=0
cs:0012 BA0A00	mov dx,000A	bx 0009	z=0
cs:0015 CD21	int 21	cx 0000	s=0
cs:0017 FEC4	inc ah	dx 0075	o=0
cs:0019 8BD6	mov dx,si	si 0075	p=1
cs:001B CD21	int 21	di 0083	a=0
cs:001D 8A5C01	mov bl,[si+01]	bp 0000	i=1
cs:0020 C6400224	mov byte ptr [bx+si+02],24	sp 0400	d=0
cs:0024 FECC	dec ah	ds 4EAE	
cs:0026 BA2200	mov dx,0022	es 4E98	
cs:0029 CD21	int 21	ss 4EB8	
cs:002B B401	mov ah,01	cs 4EA8	
cs:002D CD21	int 21	ip 0024	
cs:002F A25500	mov [0055],al		
cs:0032 FE061D00	inc byte ptr [001D]		

Below the assembly code, a memory dump is shown for the es segment:

Address	Hex Data	ASCII Data	Segment
es:01C8	00 57 00 55 00 53 00 53	W U S S	ss:0408 0000
es:01D0	00 57 00 3D 00 0B 09 49	W = d o I	ss:0406 0000
es:01D8	6E 64 69 76 69 64 20 35	ndivid 5	ss:0404 0000
es:01E0	24 00 00 0B 00 00 00 00	\$ d	ss:0402 0000
es:01E8	00 00 00 00 00 00 00 00		ss:0400 0000

Figura 7. Segmentul de date modificat după prima întrerupere 0Ah. Instrucțiunea la adresa cs:0020h, a înlocuit explicit 0Dh cu '\$' la adresa ds:bx+si+2. Registrul bx conține lungimea șirului s1.

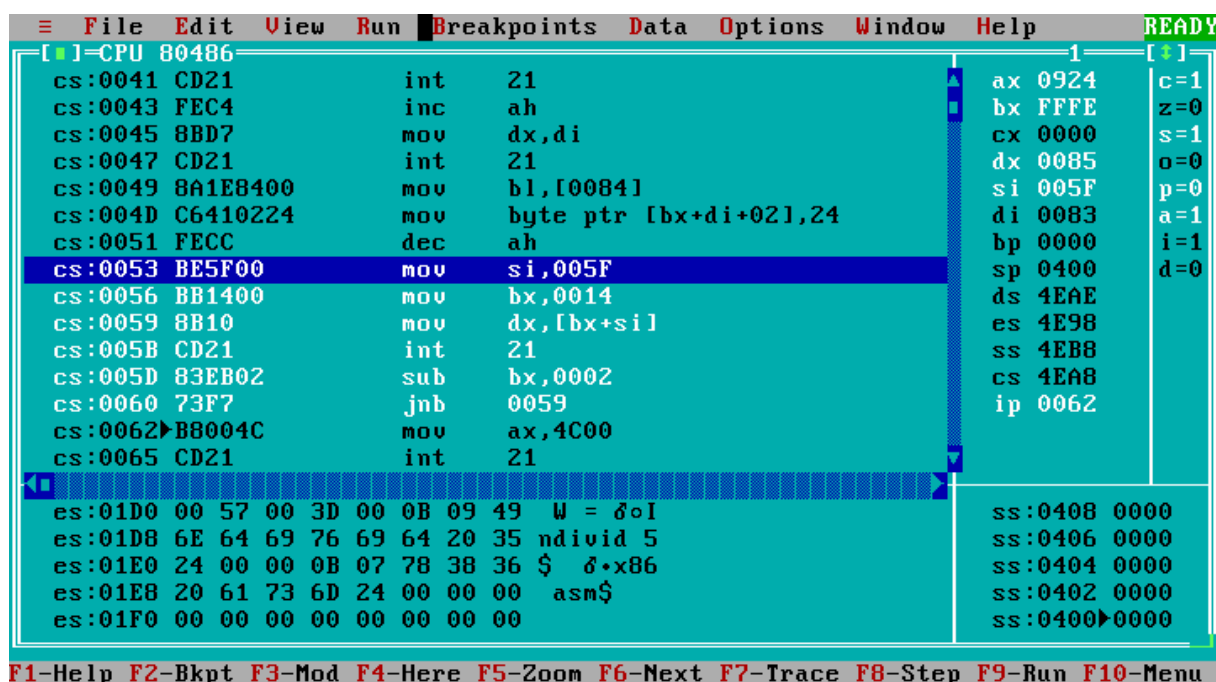


Figura 8. Bucla ce avantajează de tabelul de căutare. Tabelul conține adrese ale elementelor, care vor fi afișate pe ecran.

```
E:\>td i5
Turbo Debugger Version 3.1 Copyright (c) 1988,92 Borland International
Introduceti sirul s2:
Individ 5
Introduceti simbolul a1: ~
Introduceti sirul s3: x86 asm
Rezultatul obtinut: definit ~definit~ Individ 5x86 asm_
```

Figurile 9. Concatenarea elementelor.

Concluzie

Pe parcursul creării a acestui program, am utilizat o tehnică din programarea dinamică, ce ajută la o afișare eficientă a concatenării de elemente, prevăzute de sarcina lucrării individuale. Această tehnică se numește *tabel de căutare*, care înlocuiește computații complexe cu indexarea simplă a elementelor din tabel.

Am introdus în segmentul de cod a programului, un tabel de căutare ce conține deplasamente a elementelor ce vor fi afișate la consolă. Bucla, menționată în figura 8, iterează tabela și încarcă în registrul **dx**, adresa respectivă, pentru a afișa la ecran.

Considerând că iterația are loc de la dreapta la stânga, constă faptul că în registrul **si** este încărcată adresa începutului de tabel, și în **bx** este indicele a ultimului element a tablei. La fiecare iterație, **bx** devine decrementat cu 2 (adresa este un cuvânt), și bucla se termină când **bx** va fi mai mic decât 0 (fanionul **C** = 1).