UNIVERSITATEA DE STAT DIN MOLDOVA

FACULTATEA MATEMATICĂ ŞI INFORMATICĂ
DEPARTAMENTUL INFORMATICĂ

CEMÎRTAN CRISTIAN

Lucrarea individuală nr. 5

la disciplina ***Arhitectura Calculatoarelor și Limbaje de Asamblare***

Coordonator: Sturza Greta, lector universitar

Chişinău, 2021

Cuprins

Sarcină

Este definit şirul **s1**. Să se scrie un program ce introduce de la tastatură simbolul **a1**, şirurile **s2**, **s3** şi afişează concatenarea elementelor introduse în modul următor:

```
Introduceti sirul s2:
s2
Introduceti simbolul a1: a1
Introduceti sirul s3: s3
Rezultatul obtinut: s1bba1s1a1bbs2s3
```

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

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

; 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

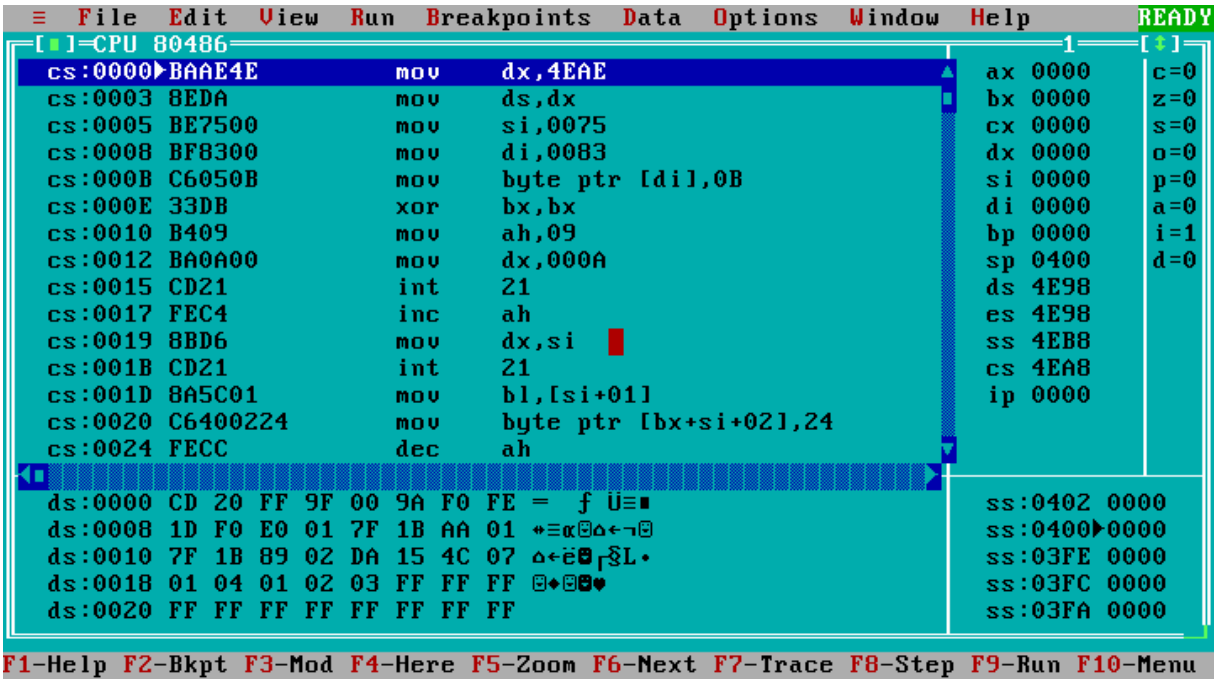

Figura 3. Depanatorul înainte de a rula programul sub format .EXE.

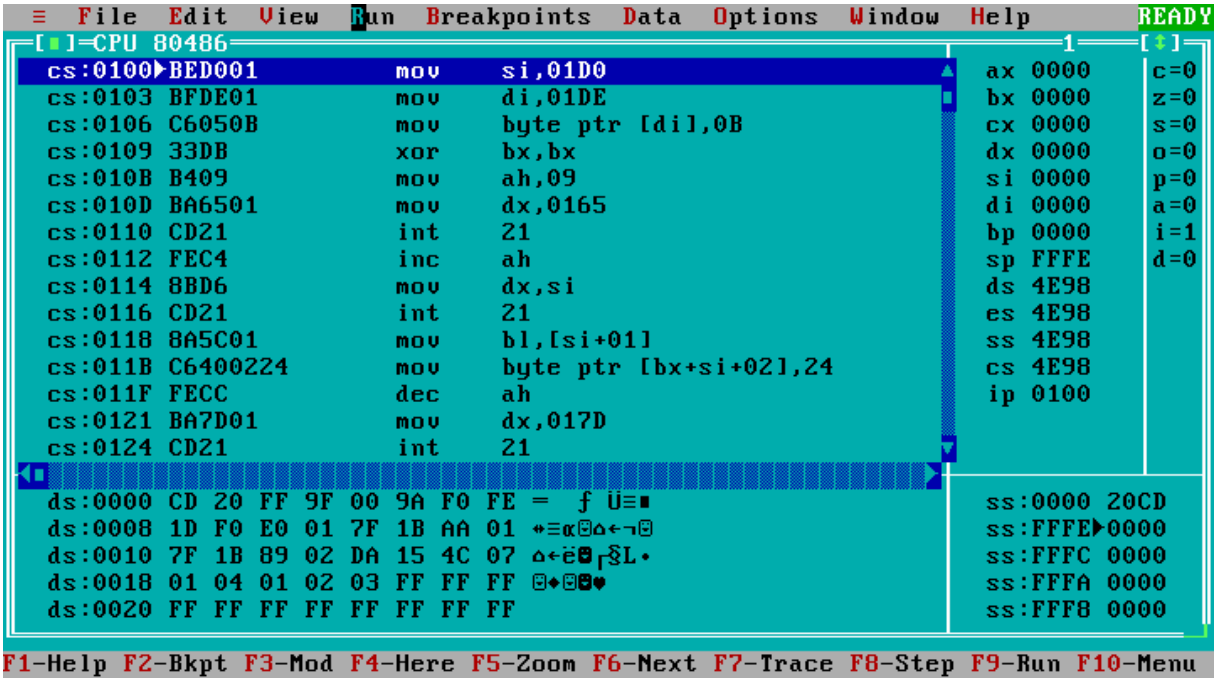

Figura 4. Depanatorul înainte de a rula programul sub format .COM. În deosebire de versiunea .EXE, nu necesită încărcarea manuală a registrului ds, deoarece modelul *small* unifică stiva, cod și date într-un singur segment.

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

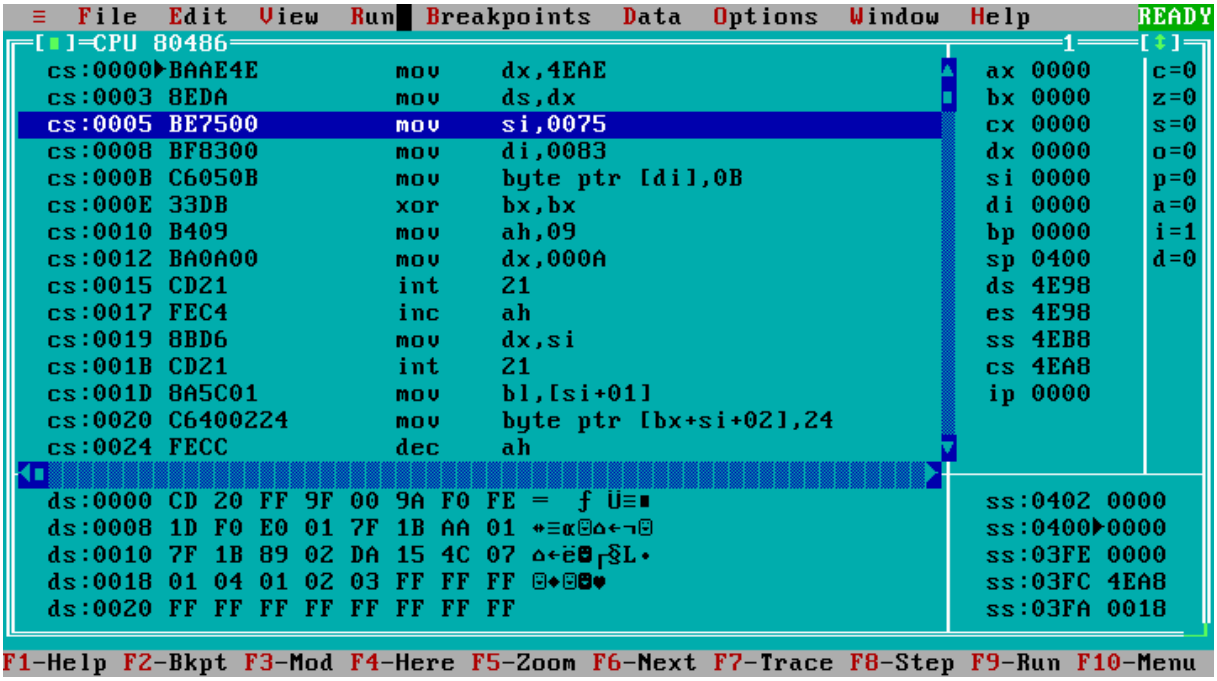

Figura 5. Registrul ds modificat explicit, după executarea instrucțiunii aflată la adresa cs:0003h. Registrul ip indică locația instrucțiunii ce va fi executată.

```
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.

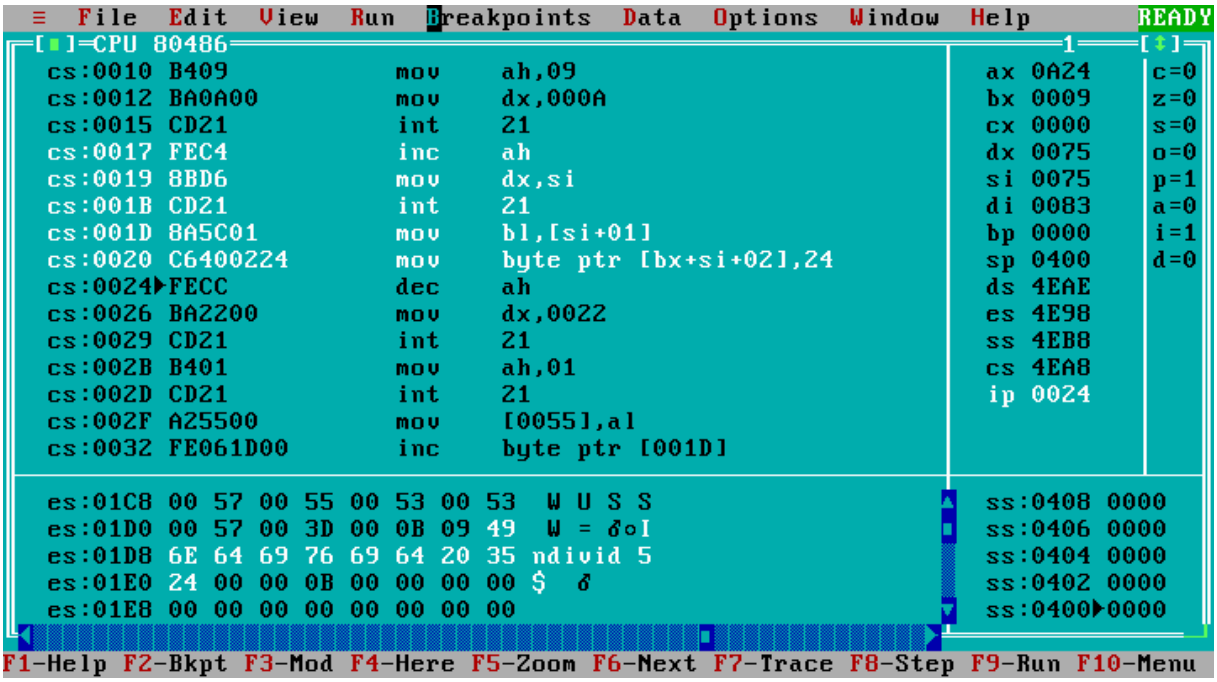

Figura 7. Segmentul de date modificat după prima întrerupere 0Ah. Instrucțiunea la adresa cs:0020h, a înlocuit explicit 0Dh cu '$' la adresa ds:bx+si+2. Registrul bx conține lungimea șirului s1.

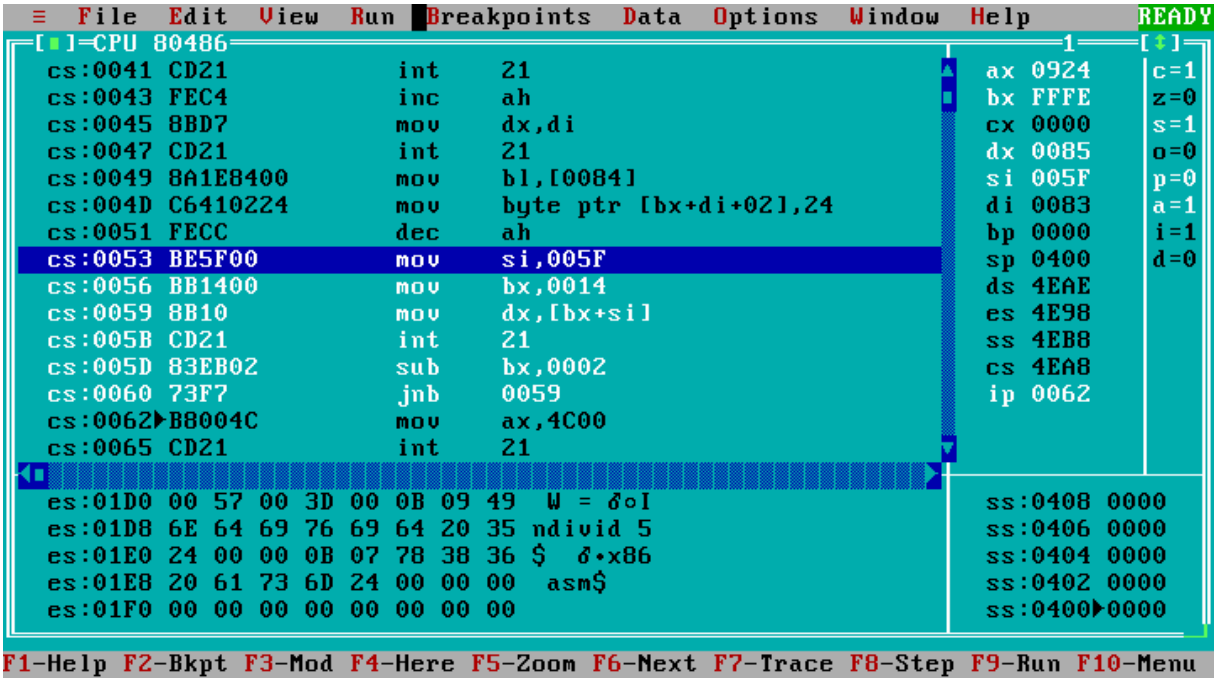

Figura 8. Bucla ce avantajează de tabelul de căutare. Tabelul conține adrese ale elementelor, care vor fi afișate pe ecran.

```
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 tabelei. 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).