Московский государственный технический университет им. Н.Э.Баумана

Защищено:

Большаков С.А.

Демонстрация ЛР:

Большаков С.А.

24 апреля 2023 г.

Отчет по лабораторной работе № 6 по курсу Системное программирование

" Ввод и распечатка параметров к.с. "

(есть ли дополнительные требования- ДА/НЕТ)

9
(количество листов)
Вариант № <**10**>

ИСПОЛНИТЕЛЬ:

студент группы **ИУ5-41Б** _____
(подпись)

Никулин Д.Д.

" "_____2023 г.

Москва, МГТУ - 2023

СОДЕРЖАНИЕ

1.Цель выполнения лабораторной работы № 6

Разработать и отладить программу на языке Ассемблер для ввода, анализа (расшифровки, фактически грамматического разбора) и распечатки параметра командной строки, которые задаются при запуске программы. Программа должна быть скомпонована в виде *.EXE-исполнимого файла. Изучить структуру PSP и способы получения в программе адреса этого блока. Распечатать заданные параметры.

2. Порядок и условия проведения работы № 6

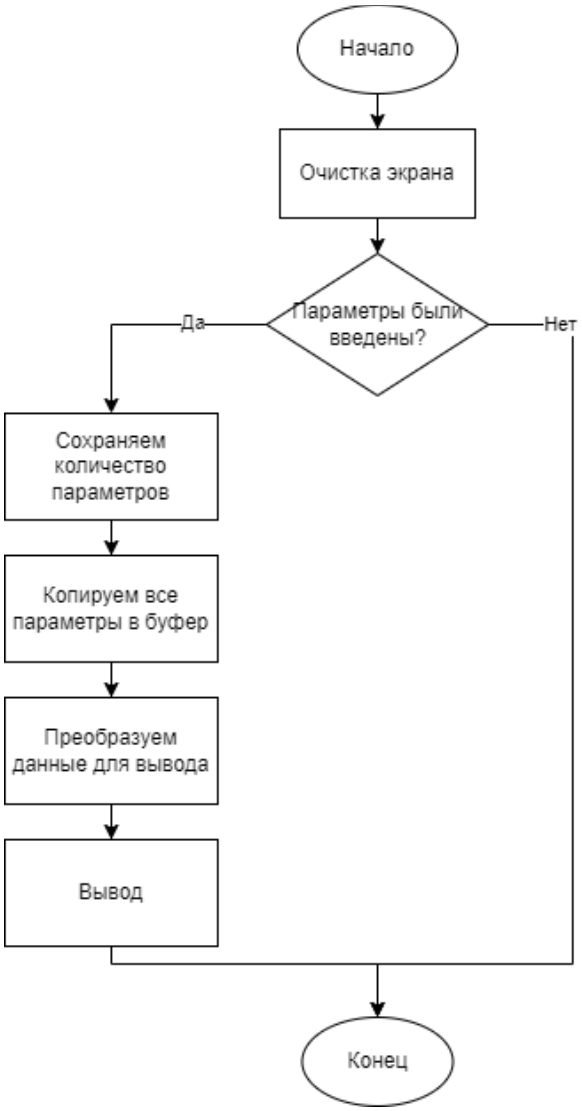
После запуска программы *.EXE список параметров (текст вводимой командной строки сохраняется в PSP программы). Доступ к PSP может быть выполнен с помощью прерывания 21h – 51h или из сегментного регистра ES после первоначального запуска программы. Поле списка параметров начинается в PSP со смещение **081h** (См. справочник). В области PSP со смещением **80H** содержится число символов введенных параметров (один байт). **Примечание**. При создании *.COM – это д.т. программы PSP располагается непосредственно в начале программы (ORG 100h – область, в которую загрузчик записывает блок PSP).

Необходимо распечатать введенные параметры все вместе (из PSP), вывести с комментарием число байт в командной строке (1 байт в HEX - 80H), а также подсчитать и вывести на консоль число параметров, текущего запуска программы которые разделены пробелом. (1 байт в HEX)

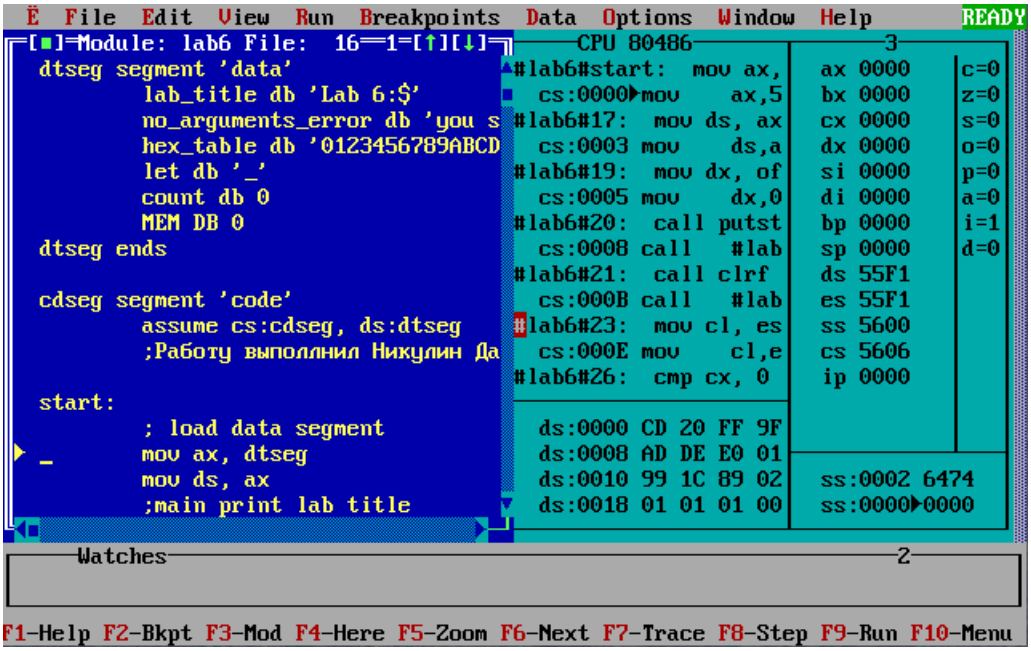
3. Описание ошибок, возникших при отладке № 6

№ п/п	Проявление ошибки	Причина ошибки	Способ устранения
1.	Error: undefined symbol … in module	Доступ ко внешней переменной, ссылка на которую не была найдена	Ввод в программе корректного сегментного регистра через ASSUME CS:… DS:…
2.	Вывод нечитаемых символов	Неправильное обращение в PSP раздел	Исправление команд, где применяется SI регистр

4.Блок-схема программы



5.Скриншот программы в TD.exe



6.Текст программы на языке Ассемблера

```
Turbo Assembler  Version 3.1            04/24/23 08:40:27            Page 1
lab6.ASM

    1    0000                           dtseg segment 'data'
    2    0000  4C 61 62 20 36 3A  24            lab_title db 'Lab 6:$'
    3    0007  79 6F 75 20 73 68  6F+           no_arguments_error     db 'you   should provide at least       2 arguments$'
    4          75 6C 64 20 70 72  6F+
    5          76 69 64 65 20 61  74+
    6          20 6C 65 61 73 74  20+
    7          32 20 61 72 67 75  6D+
    8          65 6E 74 73 24
    9    002F  30 31 32 33 34 35  36+           hex_table db '0123456789ABCDEF'
   10          37 38 39 41 42 43  44+
   11          45 46
   12    003F  5F                               let db '_'
   13    0040  00                         count db 0
   14    0041  00                         MEM DB 0
   15    0042                             dtseg ends
   16
   17    0000                             cdseg segment 'code'
   18                                           assume cs:cdseg, ds:dtseg
   19                                           ;Работу выполлнил Никулин Данила ИУ5-41Б
   20
   21    0000                             start:
   22                                           ; load data segment
   23    0000  B8 0000s                         mov ax, dtseg
   24    0003  8E D8                            mov ds, ax
   25                                           ;main print lab title
   26    0005  BA 0000r                         mov dx, offset lab_title
   27    0008  E8 005E                          call putstr;school         db 4 dup('_')
   28    000B  E8 0065                          call clrf
   29                                           ; get command line       arguments amount
   30    000E  26: 8A 0E 0080                   mov cl, es:80h
   31                                           ;mov si, 0
   32                                           ; check if         command           line argumename_messagents amount is not
zero
   33    0013  83 F9 00                         cmp cx, 0
   34    0016  74 21                            je      no_arguments
   35
   36    0018  BB 0080                          mov bx, 80h
   37    001B                                   get_arguments_cycle:
   38                                                 ; go to next cell in psp
   39    001B  43                                     inc bx
   40    001C  FE 06 0041r                            INC MEM
   41                                                 ; get    cur cell
   42    0020  26: 8A 17                              mov dl, es:bx
   43
   44    0023  80 FA 20                               cmp dl, ' '
   45    0026  75 06                                  jne jump
   46    0028  B2 20                                  mov dl, ' '
   47    002A  FE 06 0040r                            inc count
   48    002E                                         jump:
   49    002E  E8 0031                                call putch
   50
   51    0031  E2 E8                            loop get_arguments_cycle
   52    0033  E8 003D                          call clrf
   53    0036  EB 0D 90                         jmp final
   54
   55    0039                                   no_arguments:
   56    0039  BA 0007r                         mov dx, offset no_arguments_error
   57    003C  E8 002A                          call putstr
```

```
 58    003F  E8 0031                    call clrf
 59    0042  EB 01 90                   jmp final
 60
 61    0045                             final:
 62    0045  8A 16 0040r                mov dl, count
 63    0049  E8 0061                    call printdigit
 64                                     ;quit
 65    004C  E8 0024                    CALL CLRF
 66    004F  8A 1E 0040r                MOV BL, COUNT
 67    0053  8A 16 0041r                MOV DL, MEM
 68    0057  2A D3                      SUB DL, BL
 69    0059  E8 0051                    CALL printdigit
 70
 71
 72    005C  B0 00                      mov al, 0
 73    005E  B4 4C                      mov ah, 4CH
 74    0060  CD 21                      int 21H
 75
 76    0062                       putch proc
 77    0062  50                   push ax
 78    0063  B4 02                      mov ah, 02
 79    0065  CD 21                      int 21H
 80    0067  58                   pop ax
 81    0068  C3                         ret
 82    0069                       putch endp
 83
 84    0069                       putstr proc
 85    0069  B4 09                      mov ah, 09
 86    006B  CD 21                      int 21h
 87    006D  C3                         ret
 88    006E                       putstr endp
 89
 90    006E                       getch proc
 91    006E  B4 08                      mov ah, 08h
 92    0070  CD 21                      int 21h
 93    0072  C3                         ret
 94    0073                       getch endp
 95
 96    0073                       clrf proc
 97    0073  B2 0A                      mov dl, 10
 98    0075  E8 FFEA                    call putch
 99    0078  B2 0D                      mov dl, 13
100    007A  E8 FFE5                    call putch
101    007D  C3                         ret
102    007E                       clrf endp
103
104    007E                       clrscr proc
105    007E  E8 FFF2                    call clrf
106    0081  B4 00                      mov  ah , 0H
107    0083  B0 03                      mov  al , 3H
108    0085  CD 10                      int  10H
109    0087  C3                         ret
110    0088                       clrscr endp
111
112    0088                       printhex proc
113                                     ; first digit
114    0088  A0 003Fr                   mov al, let
```

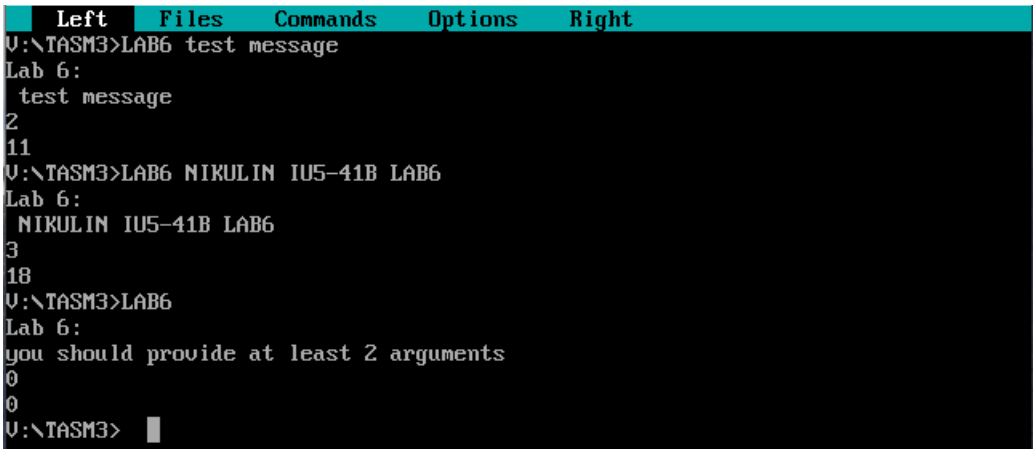
```
115  008B  D0 E8 D0 E8 D0 E8          D0+         shr al, 4
116        E8
117  0093  BB 002Fr                        lea bx, hex_table
118  0096  D7                              xlat
119  0097  8A D0                           mov dl, al
120  0099  E8 FFC6                        call putch
121
122                                        ;second digit
123  009C  A0 003Fr                        mov al, let
124  009F  24 0F                           and al, 0fh
125  00A1  D7                              xlat
126  00A2  8A D0                           mov dl, al
127  00A4  E8 FFBB                        call putch
128
129                                        ;final letter h
130  00A7  B2 68                           mov dl, 'h'
131  00A9  E8 FFB6                        call putch
132  00AC  C3                              ret
133  00AD                          printhex endp
134
135  00AD                          printdigit      proc
136  00AD  80 FA 0A                        cmp dl, 10
137  00B0  7C 12                           jl      jump1
138  00B2  B4 00                           mov ah, 0
139  00B4  8B C2                           mov ax, dx
140  00B6  B3 0A                           mov bl, 10
141  00B8  F6 F3                           div bl
142  00BA  8A D0                           mov dl, al
143  00BC  80 C2 30                        add dl, '0'
144  00BF  E8 FFA0                        call putch
145  00C2  8A D4                           mov dl, ah
146
147  00C4                                  jump1:
148  00C4  80 C2 30                        add dl, '0'
149  00C7  E8 FF98                        call putch
150
151  00CA  C3                              ret
152  00CB                          printdigit      endp
153
154  00CB                          cdseg ends
155
156                                end start
```

Symbol Table

Symbol Name	Type	Value	Cref (defined at #)
??DATE	Text	"04/24/23"	
??FILENAME	Text	"lab6 "	
??TIME	Text	"08:40:27"	
??VERSION	Number	030A	
@CPU	Text	0101H	
@CURSEG	Text	CDSEG	#1 #17
@FILENAME	Text	LAB6	
@WORDSIZE	Text	2	#1 #17
CLRF	Near	CDSEG:0073	28 52 58 65 #96 105
CLRSCR	Near	CDSEG:007E	#104
COUNT	Byte	DTSEG:0040	#13 47 62 66
FINAL	Near	CDSEG:0045	53 59 #61
GETCH	Near	CDSEG:006E	#90
GET_ARGUMENTS_CYCLE	Near	CDSEG:001B	#37 51
HEX_TABLE	Byte	DTSEG:002F	#9 117
JUMP	Near	CDSEG:002E	45 #48
JUMP1	Near	CDSEG:00C4	137 #147
LAB_TITLE	Byte	DTSEG:0000	#2 26
LET	Byte	DTSEG:003F	#12 114 123
MEM	Byte	DTSEG:0041	#14 40 67
NO_ARGUMENTS	Near	CDSEG:0039	34 #55
NO_ARGUMENTS_ERROR	Byte	DTSEG:0007	#3 56
PRINTDIGIT	Near	CDSEG:00AD	63 69 #135
PRINTHEX	Near	CDSEG:0088	#112
PUTCH	Near	CDSEG:0062	49 #76 98 100 120 127 131 144 149
PUTSTR	Near	CDSEG:0069	27 57 #84
START	Near	CDSEG:0000	#21 156

Groups & Segments	Bit Size	Align	Combine	Class	Cref (defined at #)
CDSEG	16	00CB	Para	none	CODE	#17 18
DTSEG	16	0042	Para	none	DATA	#1 18 23

7. Результаты работы программы



8.Выводы по ЛР № 6

По результату выполнения лабораторной работы №6 была разработана и отлажена программа на языке Ассемблер для обработки данных при вводе из командной строки. Было изучено строение раздела PSP, в частности обработка аргументов к.с. при запуске программы.