# Лабораторная работа №3

## Команды пересылки данных

### Вариант 1

Цель работы: изучить команды пересылки данных и особенности их применения.

### Задание к работе

1. Ознакомиться с теоретическим материалом.
2. Исследовать выполнение всех команд пересылки данных с любыми возможными типами операндов.
3. В соответствии со своим вариантом решить поставленные задачи с помощью DEBUG и TASM.
4. В отчёт включить все необходимые листинги.

### Индивидуальное задание

1. Используя любые известные команды промоделировать выполнение команды PUSH.
2. Обменять содержимое сегментных регистров DS и ES.

Отчёт

### PUSH

Команда PUSH уменьшает значение регистра-указателя стека SP/ESP на 2/4 и записывает значение источника в вершину стека по адресу SS:SP.

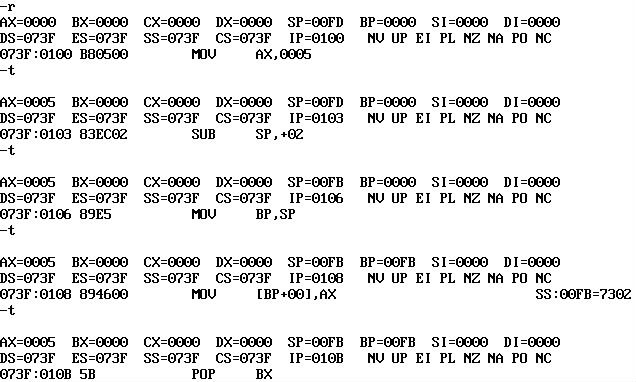
Команда PUSH имеет два операнда. Промоделируем следующую ситуацию:

PUSH 5

Команда уменьшает значение регистра-указателя стека SP/ESP на 2/4 и записывает значение 5 в вершину стека по адресу SS:SP. Выполним эти действия в DEBUG, не используя команду PUSH:



Здесь для наглядности мы воспользовались командой POP, чтобы вытолкнуть вершину стека в BX. Результат заметен в трассировке:



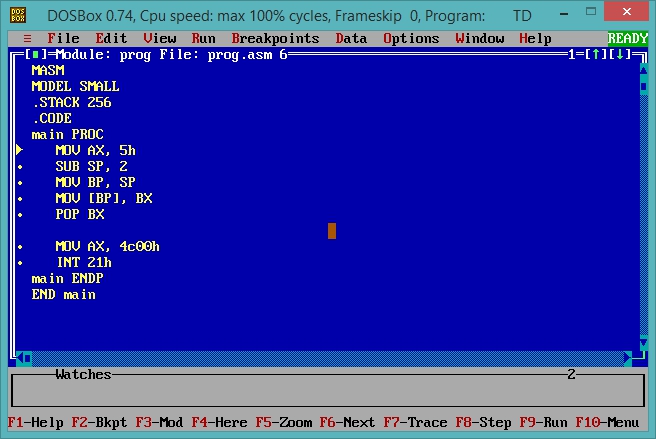


Код программы и листинг:

|  |  |
| --- | --- |
| 01 | MASM |
| 02 | MODEL SMALL |
| 03 | .STACK 100h |
| 04 | .CODE |
| 05 | main PROC |
| 06 | MOV AX, 5h |
| 07 | SUB SP, 2 |
| 08 | MOV BP, SP |
| 09 | MOV [BP], AX |
| 10 | POP BX |
| 11 |  |
| 12 | MOV AX, 4c00h |
| 13 | INT 21h |
| 14 | main ENDP |
| 15 | END main |

|  |  |
| --- | --- |
| 01 | Turbo Assembler Version 4.1 11/15/14 02:44:58 Page 1 |
| 02 | prog.ASM |
| 03 |  |
| 04 |  |
| 05 |  |
| 06 | 1 MASM |
| 07 | 2 0000 MODEL SMALL |
| 08 | 3 0000 .STACK 100h |
| 09 | 4 0000 .CODE |
| 10 | 5 0000 main PROC |
| 11 | 6 0000 B8 0005 MOV AX, 5h |
| 12 | 7 0003 83 EC 02 SUB SP, 2 |
| 13 | 8 0006 8B EC MOV BP, SP |
| 14 | 9 0008 89 5E 00 MOV [BP], BX |
| 15 | 10 000B 5B POP BX |
| 16 | 11 |
| 17 | 12 000C B8 4C00 MOV AX, 4c00h |
| 18 | 13 000F CD 21 INT 21h |
| 19 | 14 0011 main ENDP |
| 20 | 15 END main |
| 21 | Turbo Assembler Version 4.1 11/15/14 02:44:58 Page 2 |
| 22 | Symbol Table |
| 23 |  |
| 24 |  |
| 25 |  |
| 26 |  |
| 27 | Symbol Name Type Value Cref (defined at #) |
| 28 |  |
| 29 | ??DATE Text "11/15/14" |
| 30 | ??FILENAME Text "prog " |
| 31 | ??TIME Text "02:44:58" |
| 32 | ??VERSION Number 040A |
| 33 | @32BIT Text 0 #2 |
| 34 | @CODE Text \_TEXT #2 #2 #4 |
| 35 | @CODESIZE Text 0 #2 |
| 36 | @CPU Text 0101H |
| 37 | @CURSEG Text \_TEXT #4 |
| 38 | @DATA Text DGROUP #2 |
| 39 | @DATASIZE Text 0 #2 |
| 40 | @FILENAME Text PROG |
| 41 | @INTERFACE Text 000H #2 |
| 42 | @MODEL Text 2 #2 |
| 43 | @STACK Text DGROUP #2 |
| 44 | @WORDSIZE Text 2 #4 |
| 45 | MAIN Near \_TEXT:0000 #5 15 |
| 46 |  |
| 47 | Groups & Segments Bit Size Align Combine Class Cref (defined at #) |
| 48 |  |
| 49 | DGROUP Group #2 2 |
| 50 | STACK 16 0100 Para Stack STACK #3 |
| 51 | \_DATA 16 0000 Word Public DATA #2 |
| 52 | \_TEXT 16 0010 Word Public CODE #2 2 #4 4 |

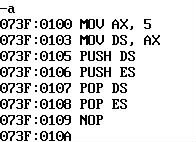
Скриншот программы:



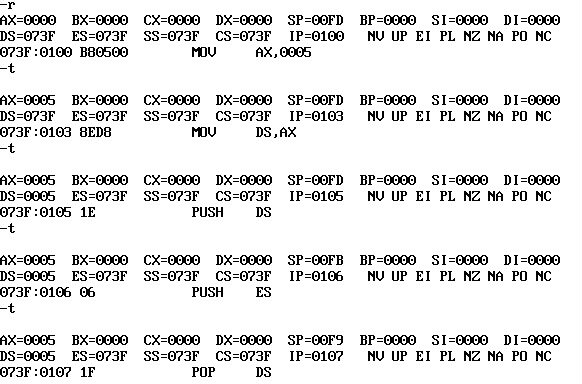
### DS и ES

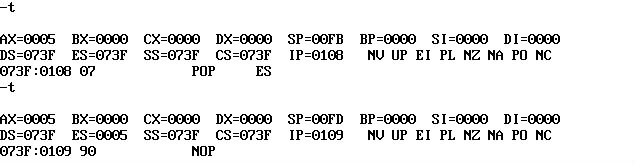
Можно разными способами обменять содержимое двух сегментных регистров — DS и ES. Промоделируем один из них — обмен при помощи стека.

Сперва запишем в один из стеков какое-нибудь тестовое значение, за которым затем можно будет следить в трассировке. После воспользуемся стеком и командами PUSH и POP. Выполним описанные действия в DEBUG:



Трассировка:





Код программы и листинг:

|  |  |
| --- | --- |
| 01 | MASM |
| 02 | MODEL SMALL |
| 03 | .STACK 100h |
| 04 | .CODE |
| 05 | main PROC |
| 06 | MOV AX, 5h |
| 07 | MOV DS, AX |
| 08 | PUSH DS |
| 09 | PUSH ES |
| 10 | POP DS |
| 11 | POP ES |
| 12 |  |
| 13 | MOV AX, 4c00h |
| 14 | INT 21h |
| 15 | main ENDP |
| 16 | END main |

|  |  |
| --- | --- |
| 01 | Turbo Assembler Version 4.1 11/15/14 04:00:14 Page 1 |
| 02 | prog.ASM |
| 03 |  |
| 04 |  |
| 05 |  |
| 06 | 1 MASM |
| 07 | 2 0000 MODEL SMALL |
| 08 | 3 0000 .STACK 100h |
| 09 | 4 0000 .CODE |
| 10 | 5 0000 main PROC |
| 11 | 6 0000 B8 0005 MOV AX, 5h |
| 12 | 7 0003 8E D8 MOV DS, AX |
| 13 | 8 0005 1E PUSH DS |
| 14 | 9 0006 06 PUSH ES |
| 15 | 10 0007 1F POP DS |
| 16 | 11 0008 07 POP ES |
| 17 | 12 |
| 18 | 13 0009 B8 4C00 MOV AX, 4c00h |
| 19 | 14 000C CD 21 INT 21h |
| 20 | 15 000E main ENDP |
| 21 | 16 END main |
| 22 | Turbo Assembler Version 4.1 11/15/14 04:00:14 Page 2 |
| 23 | Symbol Table |
| 24 |  |
| 25 |  |
| 26 |  |
| 27 |  |
| 28 | Symbol Name Type Value Cref (defined at #) |
| 29 |  |
| 30 | ??DATE Text "11/15/14" |
| 31 | ??FILENAME Text "prog " |
| 32 | ??TIME Text "04:00:14" |
| 33 | ??VERSION Number 040A |
| 34 | @32BIT Text 0 #2 |
| 35 | @CODE Text \_TEXT #2 #2 #4 |
| 36 | @CODESIZE Text 0 #2 |
| 37 | @CPU Text 0101H |
| 38 | @CURSEG Text \_TEXT #4 |
| 39 | @DATA Text DGROUP #2 |
| 40 | @DATASIZE Text 0 #2 |
| 41 | @FILENAME Text PROG |
| 42 | @INTERFACE Text 000H #2 |
| 43 | @MODEL Text 2 #2 |
| 44 | @STACK Text DGROUP #2 |
| 45 | @WORDSIZE Text 2 #4 |
| 46 | MAIN Near \_TEXT:0000 #5 15 |
| 47 |  |
| 48 | Groups & Segments Bit Size Align Combine Class Cref (defined at #) |
| 49 |  |
| 50 | DGROUP Group #2 2 |
| 51 | STACK 16 0100 Para Stack STACK #3 |
| 52 | \_DATA 16 0000 Word Public DATA #2 |
| 53 | \_TEXT 16 0010 Word Public CODE #2 2 #4 4 |

Скриншот программы:

