**Цель работы.**

Освоение средств, обеспечивающих модульное программирование.

**Задание.**

Необходимо создать основной модуль main программы, который после пуска программы осуществляет вывод на экран сообщения: “нажмите любую клавишу”, а затем вызывает call input процедуру, которая осуществляет ожидание нажатия клавиши. При нажатии клавиши процедура input вводит идентификатор нажатой клавиши и возвращает управление основному модулю main программы. В модуле main программы анализируется идентификатор нажатой клавиши. Если нажата клавиша, помеченная символом A, то модуль main программы осуществляет вызов процедуры output1, обеспечивающей печать сообщения “нажата клавиша A - взаимодействие завершено” и возвращающей управление основному модулю main программы, который завершает работу программы. Если идентификатор нажатой клавиши не соответствует символу A, то основной модуль main программы осуществляет вызов процедуры output2, которая осуществляет печать сообщения “нажмите клавишу A” и возвращает управление основному модулю main программы. Программа этого модуля в данном случае осуществляет безусловный переход (JMP) на существующую в программе команду CALL INPUT.

После разработки и отладки рассмотренной выше многомодульной программы требуется модернизировать текст модулей созданной программы таким образом, чтобы обеспечить демонстрацию передачи параметров процедурам через стек. Модернизированную программу необходимо отладить и составить отчет о проделанной работе.

Директива TITLE.

Директива TITLE текст, представляющий заголовок листинга, инициирует печать на каждой странице листинга заголовка, включающего имя модуля (6 символов) и описание (60 символов) его назначения. Листинг представляет собой текстовый файл, содержащий описание исходной ассемблерной программы, код каждой ассемблированной команды, список используемых меток, описание перекрестных ссылок, сегментов и групп сегментов.

Директивы EXTRN и PUBLIC.

Необходимость использования директив EXTRN и PUBLIC возникает при описании взаимодействующих программных модулей.

Типы для переменных: byte, word, dword.

Типы для меток:

NEAR – указывает на размещение метки внутри сегмента;

FAR – указывает на размещение метки за пределами данного сегмента;

SHORT – метки указанного типа используется в инструкциях перехода для задания адреса назначения. Пределы перехода от текущей команды ограничены величинами +127 и -128 байт.

Процедура, или подпрограмма – это основная функциональная единица декомпозиции некоторой задачи. Процедура представляет собой группу команд для решения конкретной подзадачи и обладает средствами получения управления из точки вызова задачи более высокого уровня и возврата управления в эту точку. В простейшем случае программа может состоять из одной процедуры. Другими словами, процедуру можно определить как правильным образом оформленную совокупность команд, которая, будучи однократно описана, при необходимости может быть вызвана в любом месте программы.

Директива описания процедуры.

Процедура – это поименованная часть программы. Используется для структурирования программы. Вызывается с помощью команды: call имя\_процедуры. При работе с процедурами необходимо следить за состоянием используемого стека. Для оформления процедуры используется директива PROC.

1. На первом этапе необходимо создать с помощью текстового редактора четыре программных модуля: MAIN, INPUT, OUTPUT1, OUTPUT2 и сохранить их в четырех соответствующих файлах с добавлением к именам так называемого расширения asm.
2. На втором этапе для каждого исходного модуля необходимо построить объектный модуль. Для достижения этой цели необходимо четырежды задать и выполнить следующую командную строку, изменяя nameprog на соответствующее имя.

>TASM/Z/ZI/L nameprog.asm

1. На третьем этапе для построения исполняемого модуля с возможностью дальнейшей отладки создаваемой программы с помощью программы TD.EXE необходимо задать следующую командную строку.

>TLINK/I/M/L/S/V main.obj vvod.obj vivod.obj,name.exe,name.map

1. На четвертом этапе необходимо осуществить отладку созданной программы, используя программу TD.EXE.

**Код программы.**

**LAB6.asm**

TITLE LAB6

.Model Small; задание модели "Small"

.STACK 100h; сегмент стека в 100h байт

.Code; сегмент кода

EXTRN INPUT:far, OUTPUT1:far, OUTPUT2:far

START:

call OUTPUT2; вызов процедуры OUTPUT2 для сообщения пользователю о необходимости ввода символа

CALL\_INPUT:

call INPUT; ВЫЗОВ процедуры INPUT

cmp al, 41h; сравнение введённого символа с кодом символа А

je CALL\_OUTPUT1; если введённы символ совпал с кодом символа А, то переходим к вызову процедуры OUTPUT1

call OUTPUT2; вызов процедуры OUTPUT2 для сообщения пользователю о необходимости ввода корректного символа

jmp CALL\_INPUT; Переход к метке повторного вызова процедуры INPUT

CALL\_OUTPUT1:

call OUTPUT1; вызов процедуры OUTPUT1, значит введённый символ соответствует заданному и переходим к заключительным действиям

int 21h

mov ax,4c00h

int 21h

END START

**Input.asm**

TITLE INPUT

PUBLIC INPUT ; Объявляем процедуру INPUT публичной для работы других модулей

.model small

.stack 100h

.data

.code

INPUT PROC far

mov ax, @data ; указываем на начало данных

mov ds, ax ; пересылка в DS адреса начала сегмента данных

mov ah, 1 ; вызываем функцию ввода символа

int 21h ; прерывание для отработки функции

ret ; возврат к текущему месту в программе

INPUT endp

end

**Output1.asm**

TITLE OUTPUT1

PUBLIC OUTPUT1; Объявляем процедуру OUTPUT1 публичной для работы других модулей

.model small; задание модели "Small"

.stack 100h; сегмент стека в 100h байт

.data; сегмент данных

mes1 db 10,13, 'Key A pressed - communication ended. $'

.code; сегмент кода

OUTPUT1 PROC far

mov ax,@data; указываем на начало данных

mov ds,ax ; пересылка в DS адреса начала сегмента данных

mov ah, 09h ; AH = 9 (прерывание для вывода)

lea dx,mes1; загрузка в регистр адрес mes1

int 21h ; вывод сообщения на экран

ret; возврат управления из процедуры вызывающей программе

OUTPUT1 endp

End

**Output2.asm**

TITLE OUTPUT2

PUBLIC OUTPUT2; Объявляем процедуру OUTPUT2 публичной для работы других модулей

.model small; задание модели "Small"

.stack 100h; сегмент стека в 100h байт

.data; сегмент данных

mes2 db 10,13, 'Press the A key: $'

.code; сегмент кода

OUTPUT2 PROC far

mov ax,@data; указываем на начало данных

mov ds,ax ; пересылка в DS адреса начала сегмента данных

mov ah, 09h ; AH = 9 (прерывание для вывода)

lea dx,mes2; загрузка в регистр адрес mes2

int 21h ; вывод на экран сообщения

ret; возврат управления из процедуры вызывающей программе

OUTPUT2 endp

end

Тестирование программы:

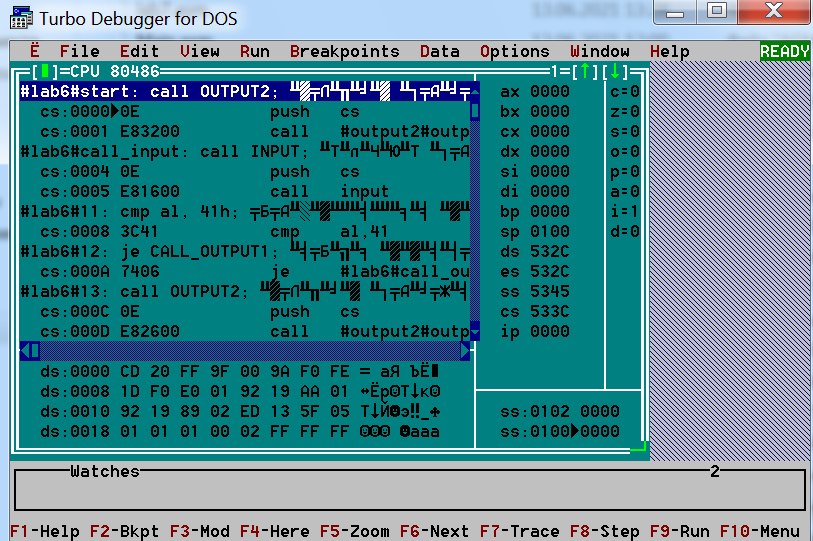


Рис. 1.1 – Программа до отладки

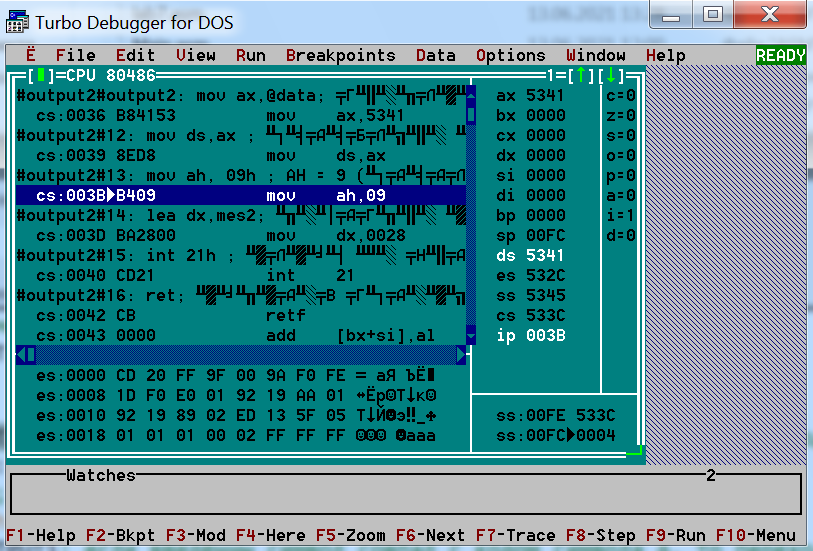


Рис. 1.2 – вызов процедуры OUTPUT2

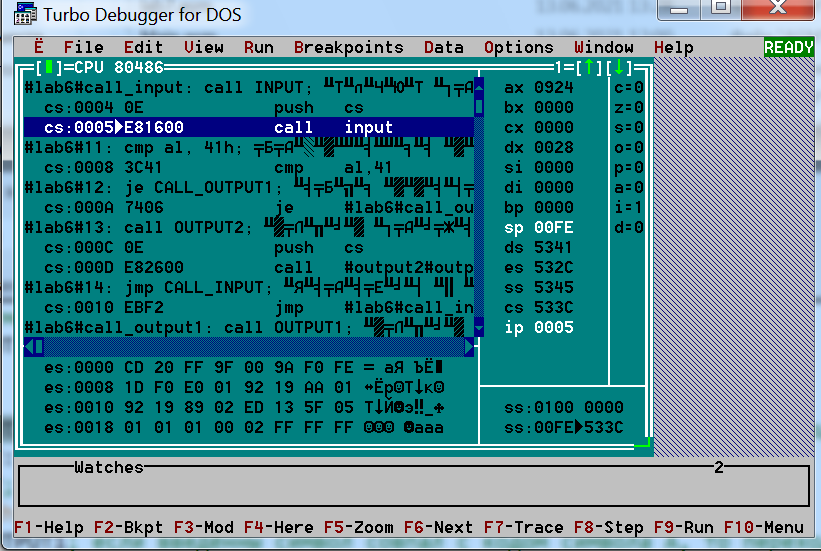


Рис. 1.3 – Вызов процедуры INPUT после OUTPUT2

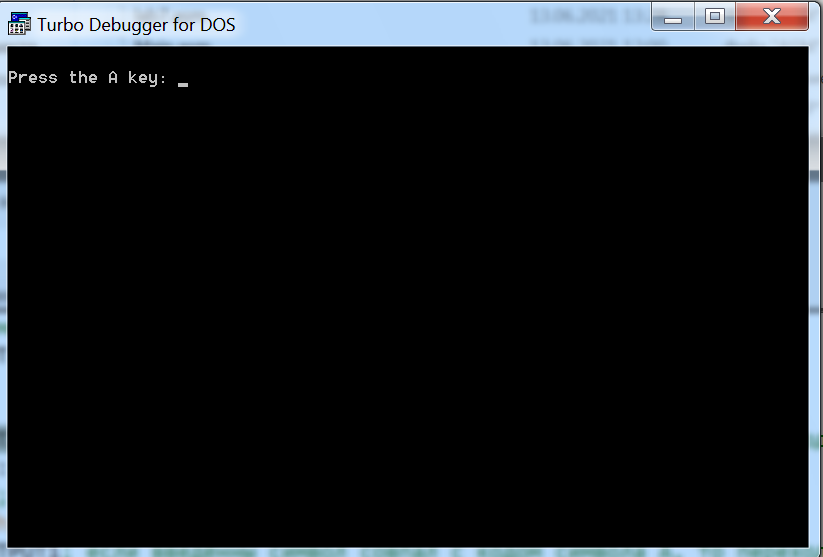


Рис. 1.4 – Ожидание ввода

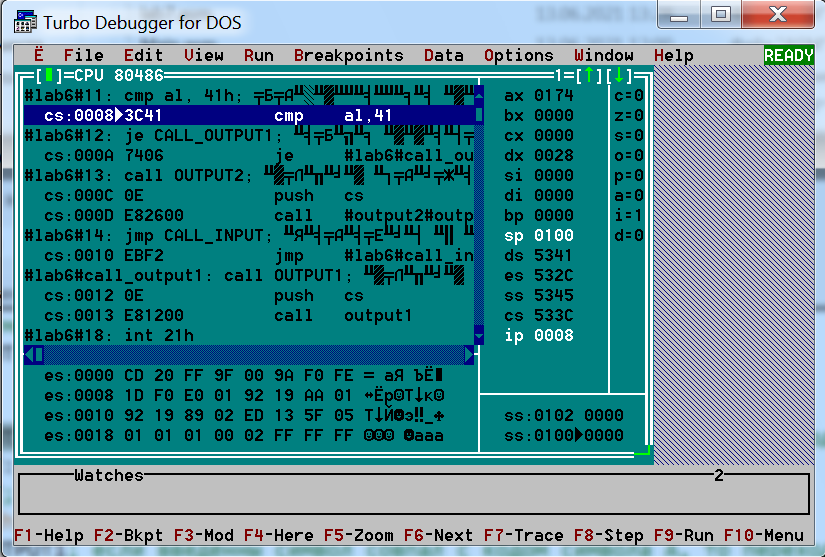


Рис. 1.5 сравнение введённого символа со значением 41h, которое соответствует символу А из таблицы ASCII

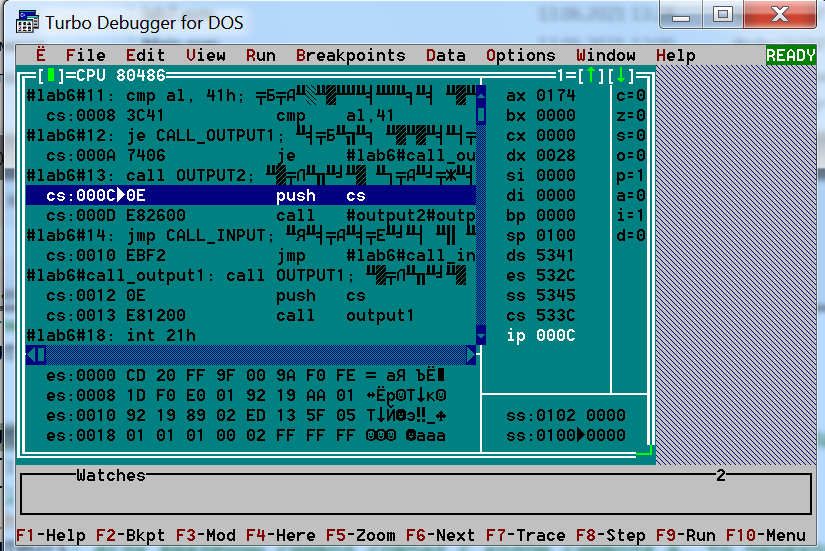


Рис. 1.6 Код введенного символа не совпал с кодом символа А

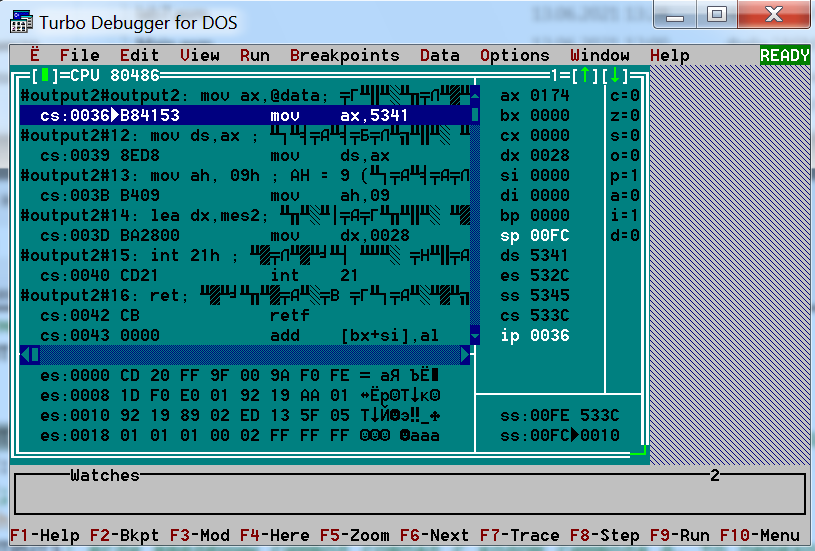


Рис.1.7 вызов процедуры OUTPUT2, поскольку символ не совпал

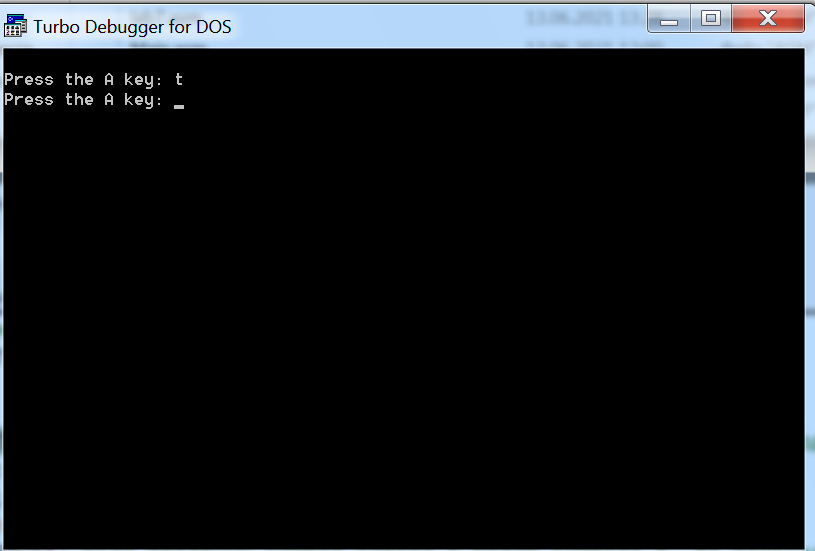
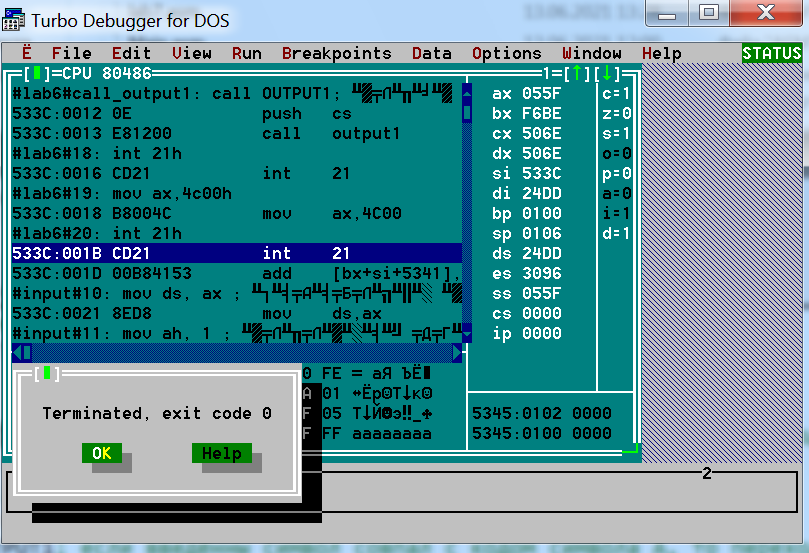


Рис.1.8 Ожидание ввода пользователя после вызова процедуры OUTPUT2, поскольку символ не совпал



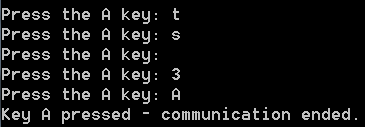


Рис. 1.9-10 Программа после отладки

Введем верное решение не стразу, чтобы посмотреть, как работает программа.

При неверном символе ожидается, что программа предложит новый ввод. Как только пользователь введет верный символ, программа должна закончить работу.

Как мы видим, программа отработала верно, так как при неверном вводе, она просит ввести новый символ, а при верном вводе, завершает свою работу.

**Модернизируем текст модулей созданной программы таким образом, чтобы обеспечить демонстрацию передачи параметров процедурам через стек.**

**LAB6.asm**

TITLE LAB6

.Model Small; задание модели "Small"

.STACK 100h; сегмент стека в 100h байт

.data; сегмент данных

PressedKey db ?

mes1 db 10,13, 'Key A pressed - communication ended. $'

mes2 db 10,13, 'Press the A key: $'

.Code; сегмент кода

EXTRN INPUT:far, OUTPUT1:far, OUTPUT2:far

START:

mov ax,@data ; указываем на начало данных

mov ds,ax ; пересылка в DS адреса начала сегмента данных

lea dx,mes2

push dx

call OUTPUT2; вызов процедуры OUTPUT2 для сообщения пользователю о необходимости ввода символа

add sp,2

CALL\_INPUT:

lea dx,PressedKey

push dx

call INPUT; ВЫЗОВ процедуры INPUT

add sp,2 ;

lea di,PressedKey

mov al,[di]

cmp al, 41h; сравнение введённого символа с кодом символа А

je CALL\_OUTPUT1; если введённы символ совпал с кодом символа А, то переходим к вызову процедуры OUTPUT1

lea dx,mes2

push dx

CALL OUTPUT2; вызов процедуры OUTPUT2 для сообщения пользователю о необходимости ввода корректного символа

add sp,2

jmp CALL\_INPUT; Переход к метке повторного вызова процедуры INPUT

CALL\_OUTPUT1:

lea dx,mes1

push dx

call OUTPUT1; вызов процедуры OUTPUT1, значит введённый символ соответствует заданному и переходим к заключительным действиям

add sp,2

int 21h

mov ax,4c00h

int 21h

END START

**Input.asm**

TITLE INPUT

PUBLIC INPUT ; Объявляем процедуру INPUT публичной для работы других модулей

.model small

.stack 100h

.code

INPUT PROC far

push bp

push ax

push di

mov bp, sp

mov ah, 1 ; вызываем функцию ввода символа

int 21h ; прерывание для отработки функции

mov di,[bp+10]

mov [di],al

pop di

pop ax

pop bp

ret ; возврат к текущему месту в программе

INPUT endp

end

**Output1.asm**

TITLE OUTPUT1

PUBLIC OUTPUT1; Объявляем процедуру OUTPUT1 публичной для работы других модулей

.model small; задание модели "Small"

.stack 100h; сегмент стека в 100h байт

.code; сегмент кода

OUTPUT1 PROC far

push bp

push ax

push dx

mov bp,sp

mov ah, 09h ; AH = 9 (прерывание для вывода)

mov dx,[bp + 10]

int 21h ; вывод сообщения на экран

pop dx

pop ax

pop bp

ret; возврат управления из процедуры вызывающей программе

OUTPUT1 endp

End

**Output2.asm**

; Дополнительный исходный файл

TITLE OUTPUT2

PUBLIC OUTPUT2; Объявляем процедуру OUTPUT2 публичной для работы других модулей

.model small; задание модели "Small"

.stack 100h; сегмент стека в 100h байт

.code; сегмент кода

OUTPUT2 PROC far

push bp

push ax

push dx

mov bp,sp

mov ah, 09h ; AH = 9 (прерывание для вывода)

mov dx,[bp + 10]

int 21h ; вывод на экран сообщения

pop dx

pop ax

pop bp

ret; возврат управления из процедуры вызывающей программе

OUTPUT2 endp

end

Продемонстрируем работу модернизированной программы:

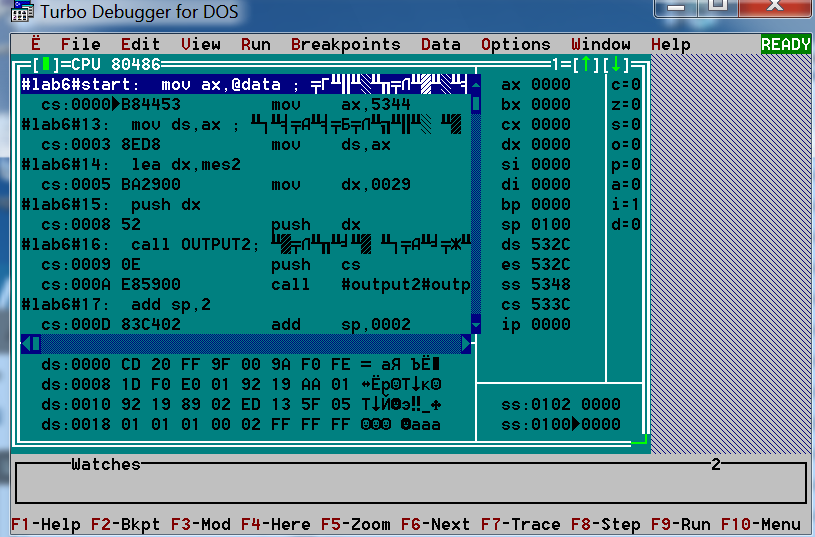
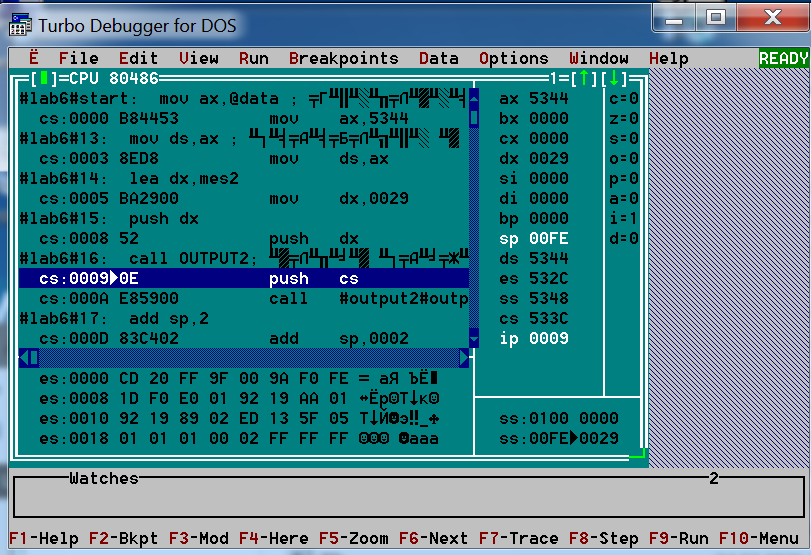
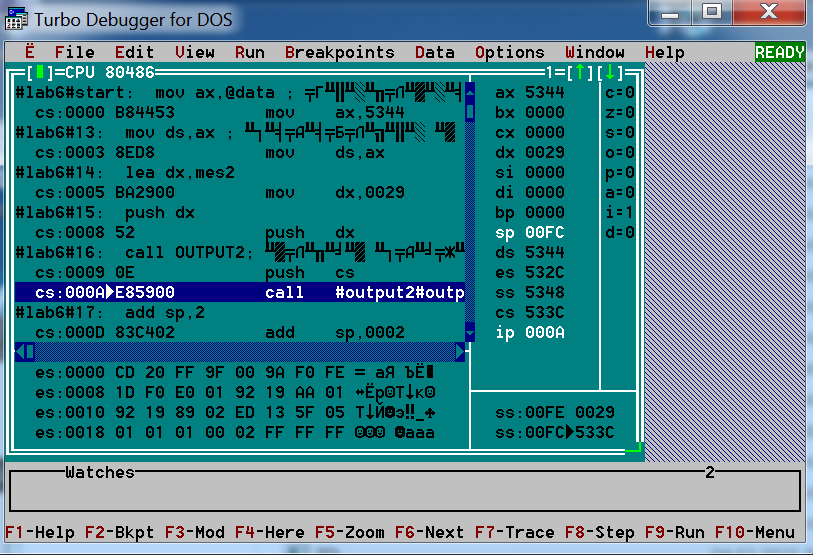
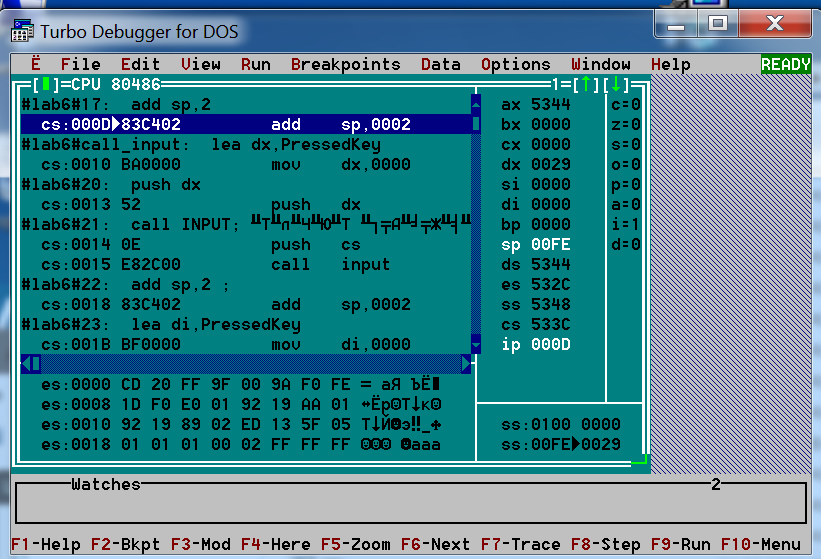


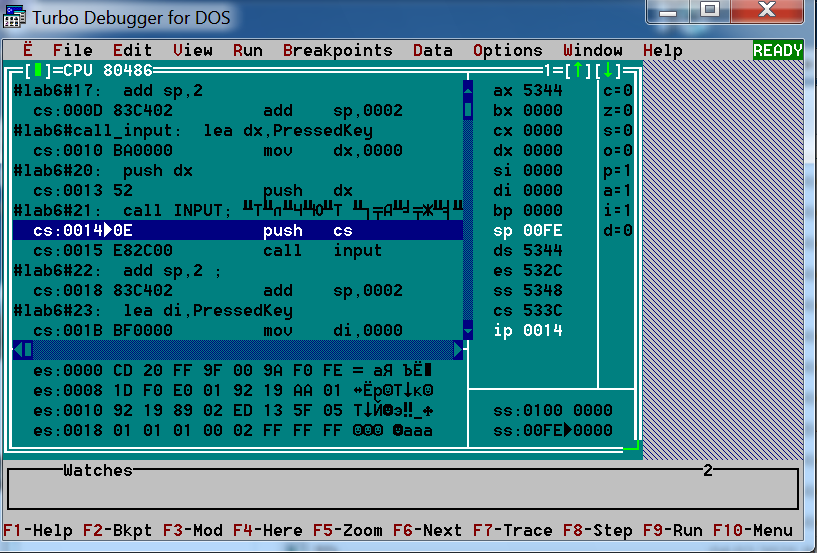
Рис. 2.1 – Программа до отладки

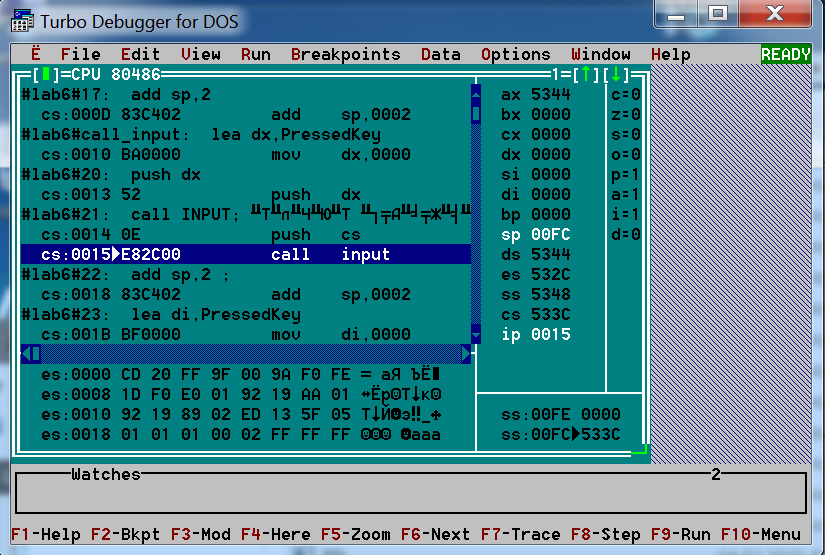
Следующие скрины показывают передачу параметров в стек.

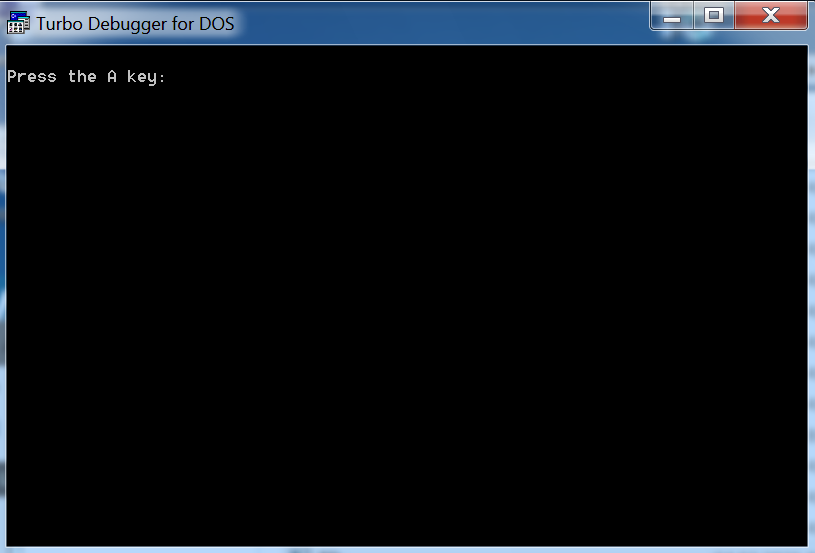


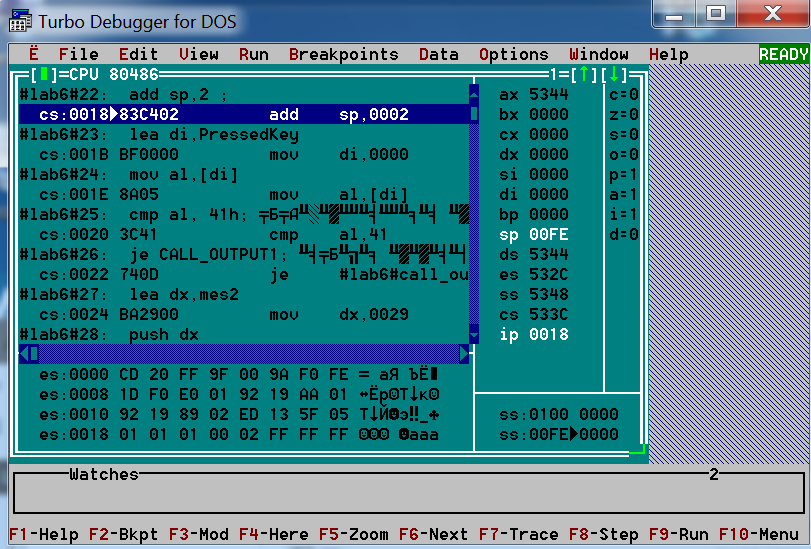


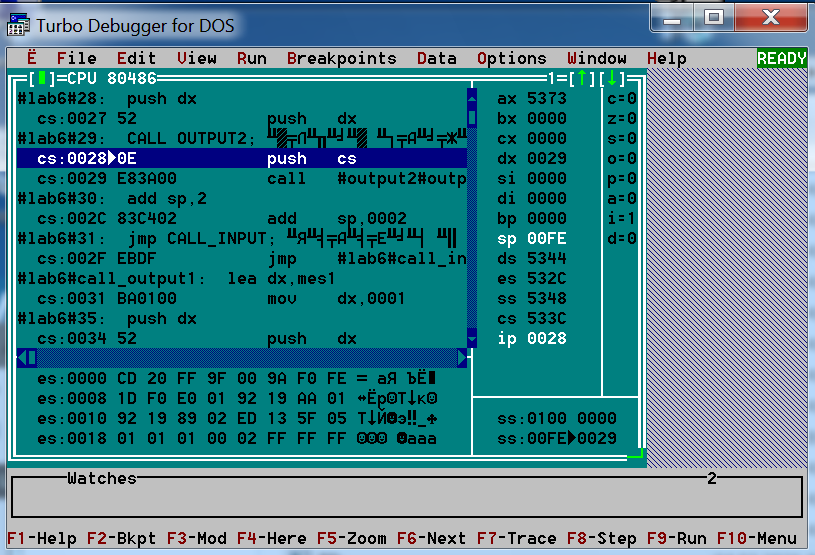


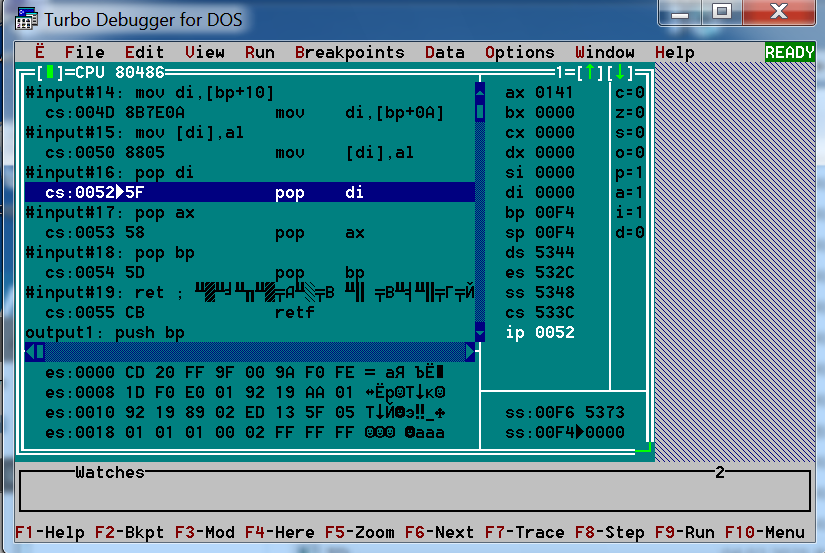


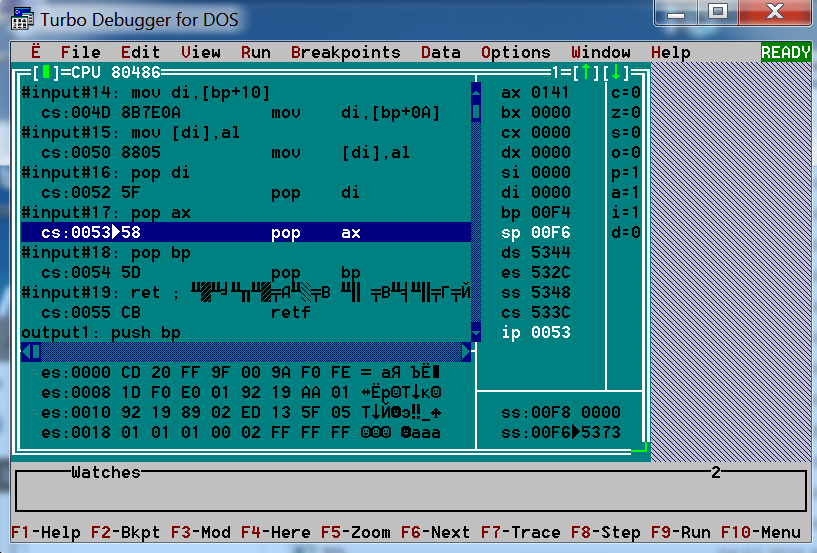


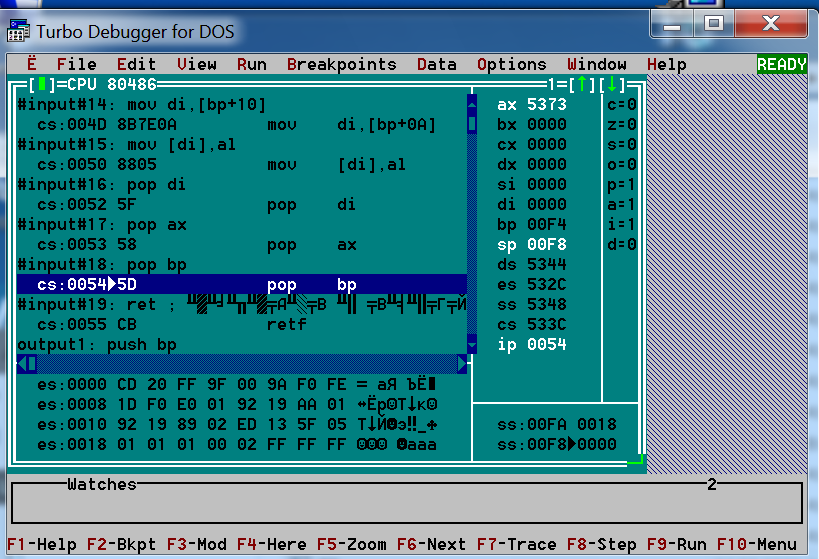


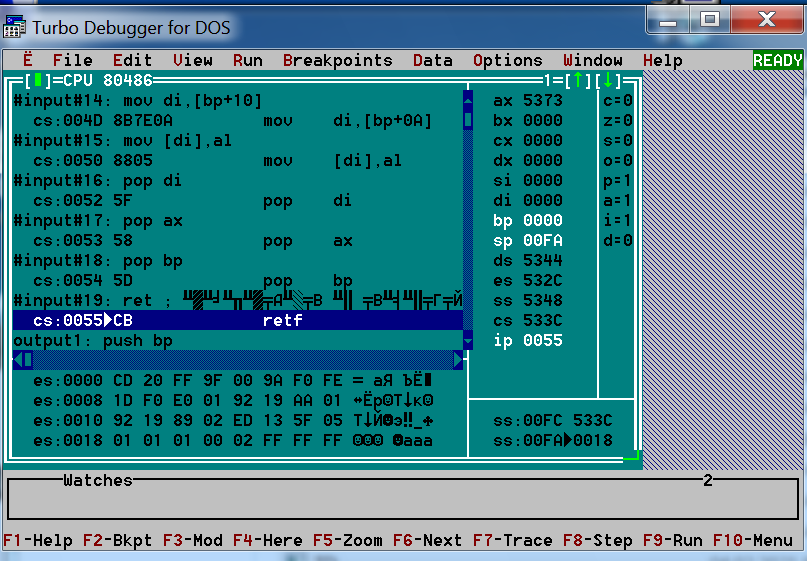


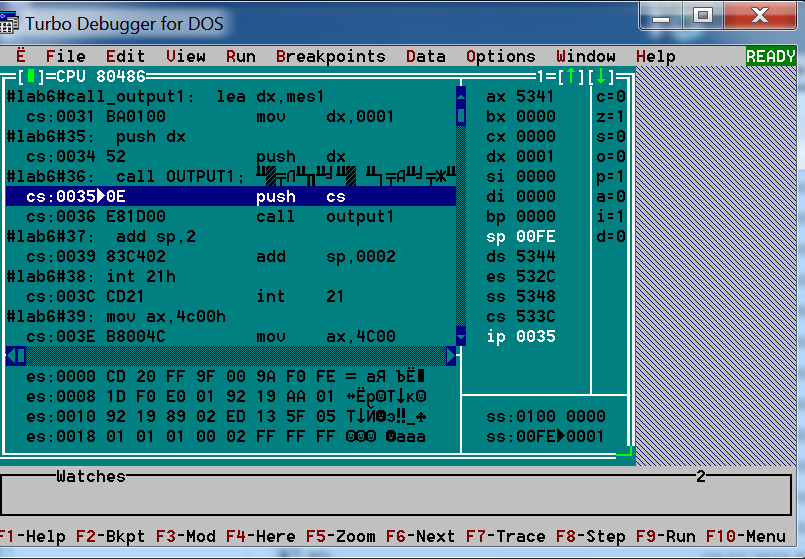


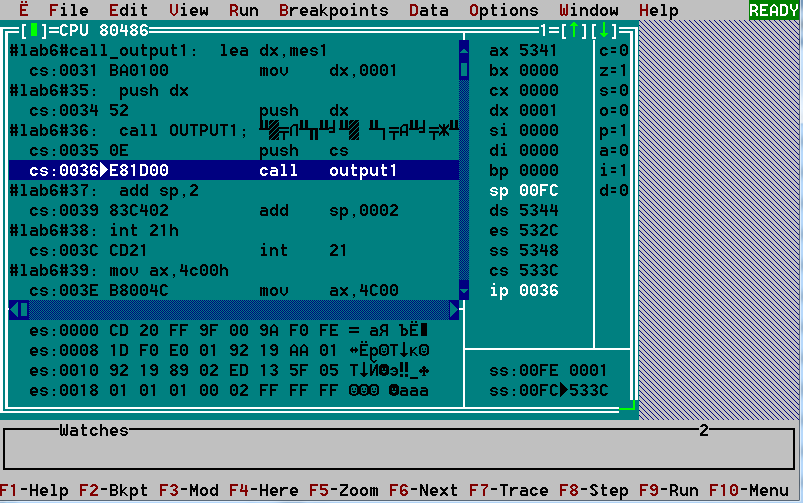


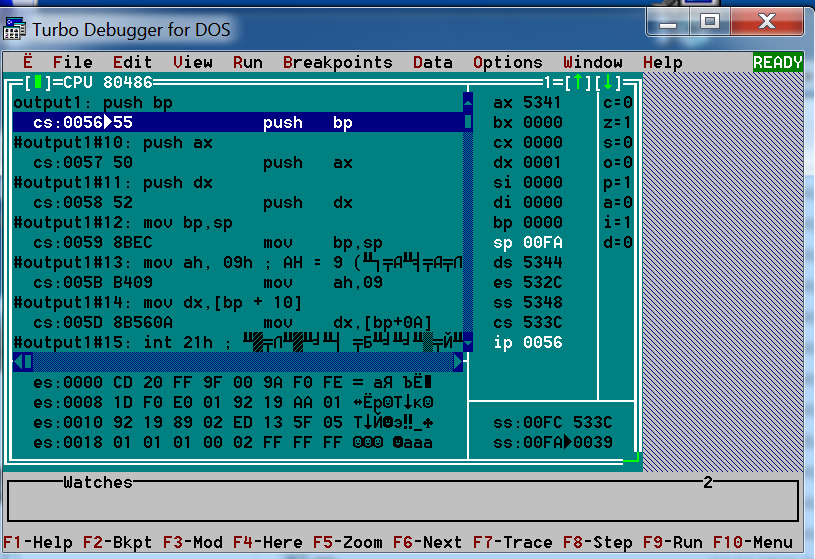


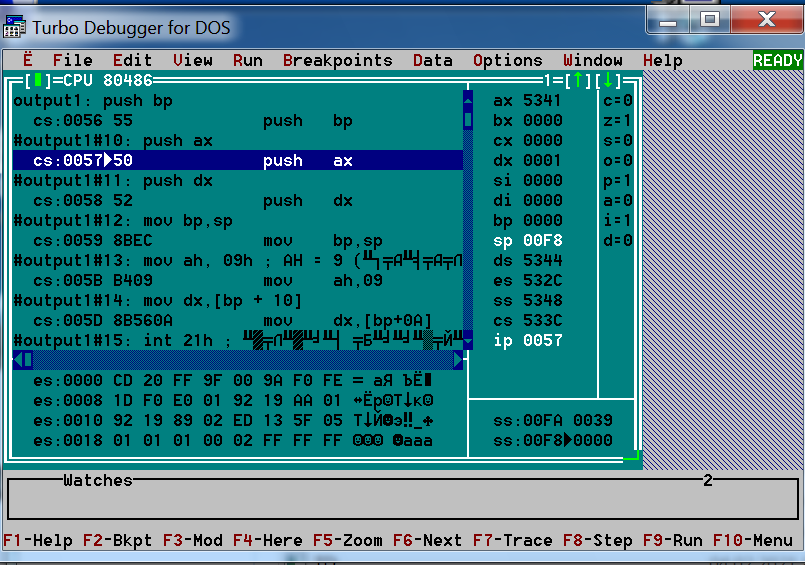


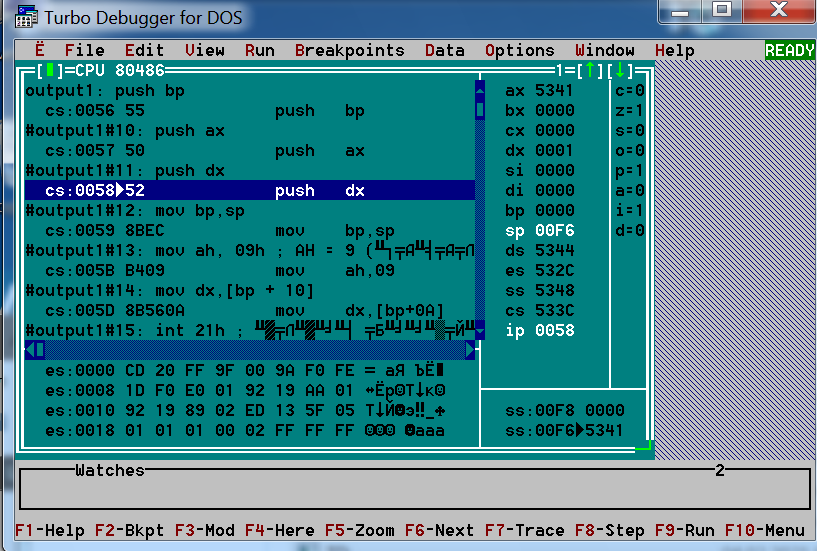


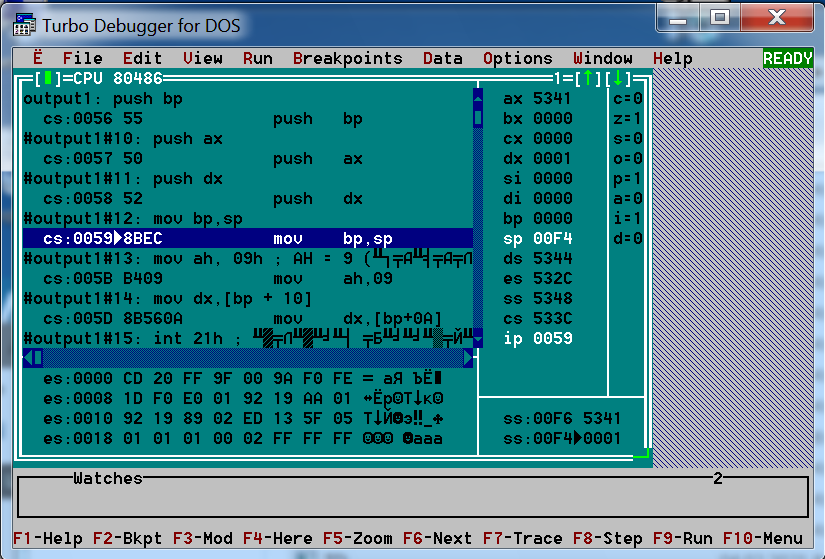


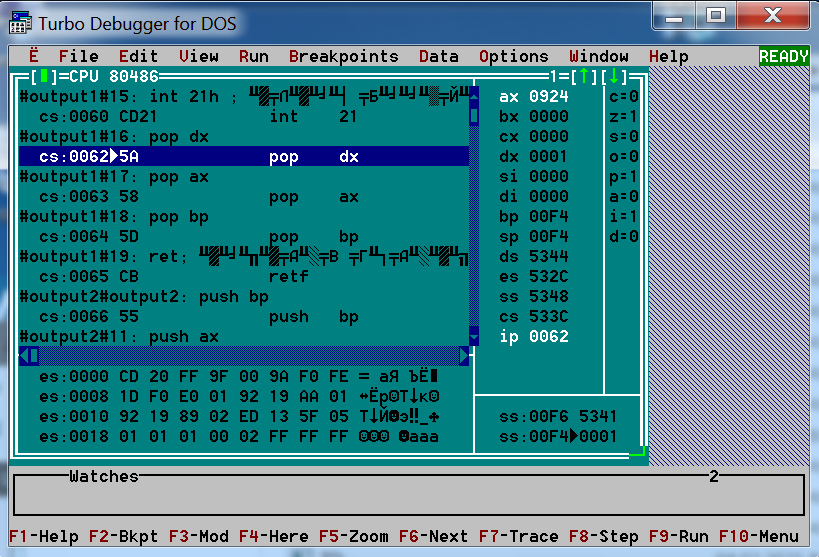


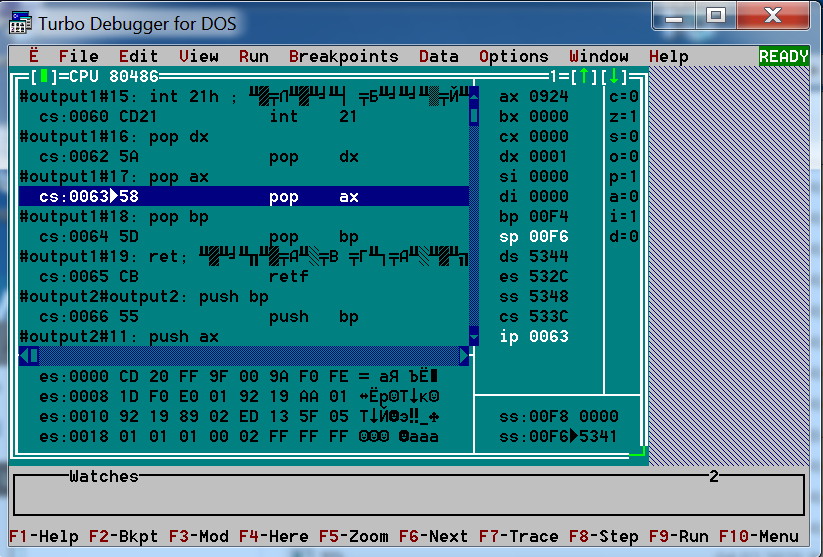


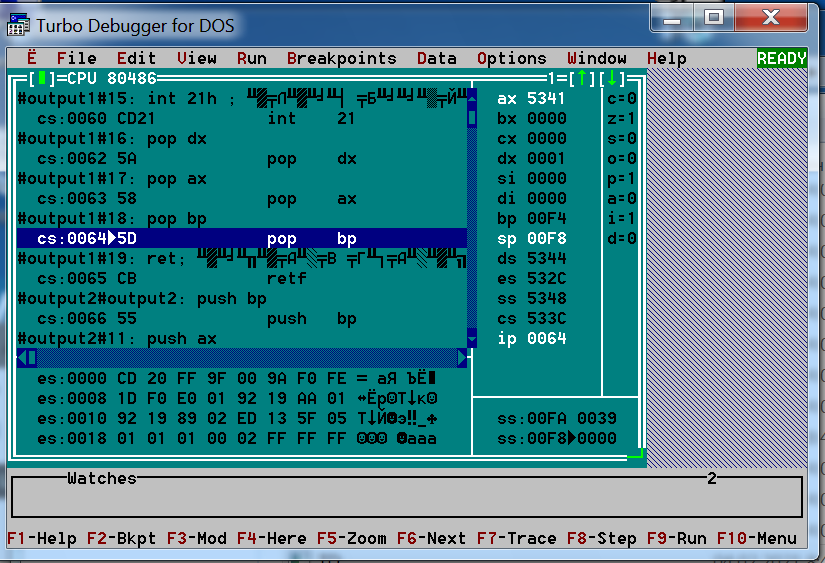


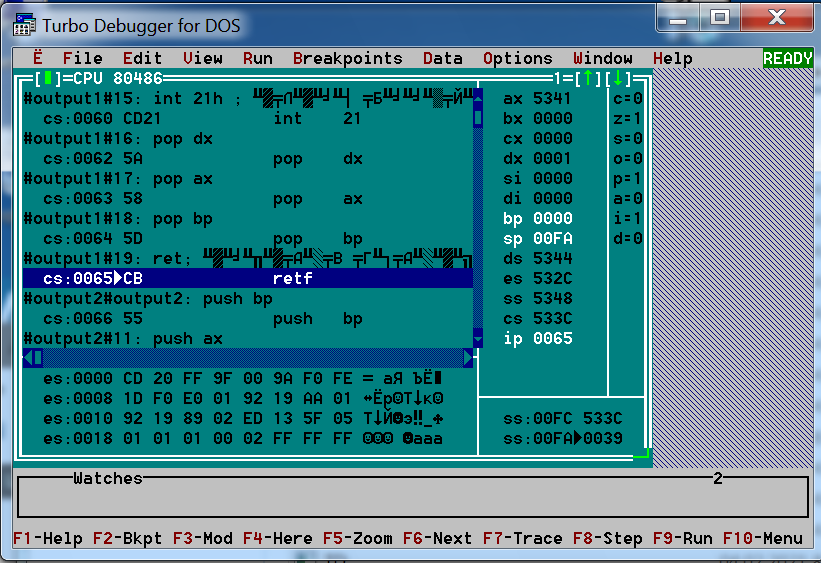


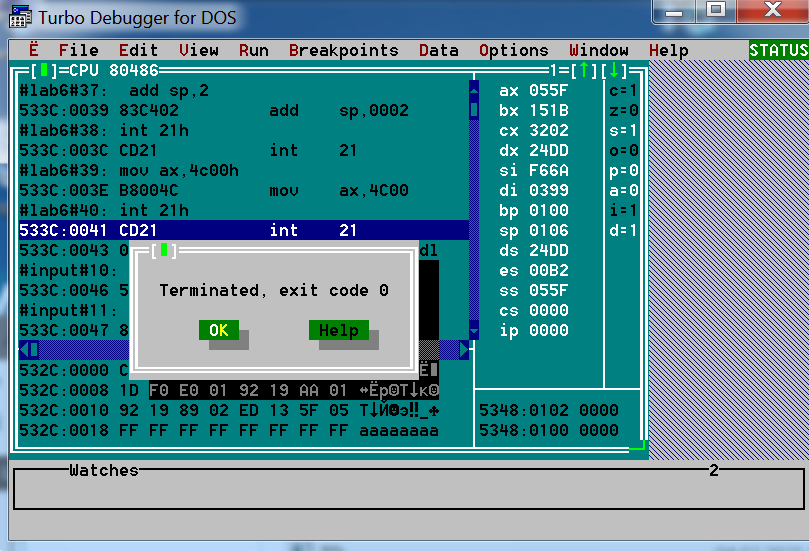














Программа отработала корректно, даже несмотря на то, что были введены неверные символы.

**ВЫВОД**

В ходе выполнения данной лабораторной работы я освоила: навык написания и компиляции программы, состоящей из нескольких программных модулей, использование директив public и extrn, способы передачи параметров процедурам через регистры и через стек, навык составления тест-кейсов для проверки корректной работы программы, навык документирования проделанной работы.