Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧЕТ**

по лабораторной работе №4

по дисциплине «ЭВМ и переферийные устройства»

на тему «Работа процессора с подпрограммами»

Выполнили:

студенты группы 22ВВП1

Хоссейни Нежад С.А.С.М.

Захаров А. С.

Приняли:

Никишин К.И.

Патунин Д.В.

Пенза 2025

**Цель работы**

Изучение особенностей выполнения команд работы с подпрограммами и передачи параметров в подпрограммы.

**Задание**

**Задание 1:** Составить программу, выполняющую обработку массива, аналогично заданию предыдущей работы, но оформленной в виде "ближней" процедуры, параметры в которую передаются через регистры.

**Задание 2:** Составить программу, выполняющую обработку массива, аналогично заданию предыдущей работы, но оформленной в виде "ближней" процедуры, параметры в которую передаются через стек.

****

**Ход работы**

**Задание 1**

**Листинг:**

stseg segment para stack

dw 16 dup(?)

stseg ends

dseg segment para

x dw 1, 0, 1, 2, 3, 0, 4, 5

y dw ?

dseg ends

cseg segment para

main proc far

; Инициализация сегмента данных

assume cs:cseg, ds:dseg, ss:stseg

push ds

mov ax, 0

push ax

mov ax, dseg

mov ds, ax

mov bx, offset x

mov cx, 8

; Инициализируем y значением 0

mov ax, 0

mov y, ax

call is\_there\_null

end\_proc:

ret

main endp

is\_there\_null proc near

loop\_start:

mov dx, [bx] ; Загружаем текущий элемент массива

cmp dx, 0

je set\_y\_to\_one ; Если элемент равен 0, то переходим в set\_y\_to\_one

add bx, 2 ; Переход к следующему элементу (одно слово = 2 байта)

loop loop\_start ; Уменьшаем CX и повторяем цикл, если CX не равен 0

ret ; Возврат в основную программу

set\_y\_to\_one:

mov ax, 1

mov y, ax ; Присваиваем y значение 1

ret ; Возврат в основную программу

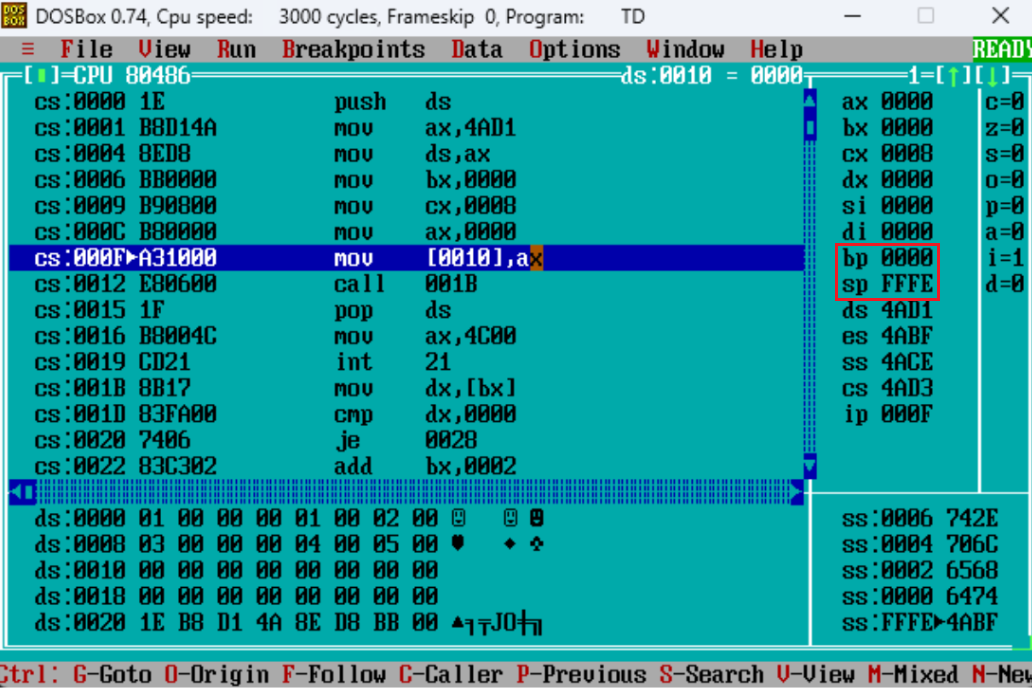
is\_there\_null endp

cseg ends

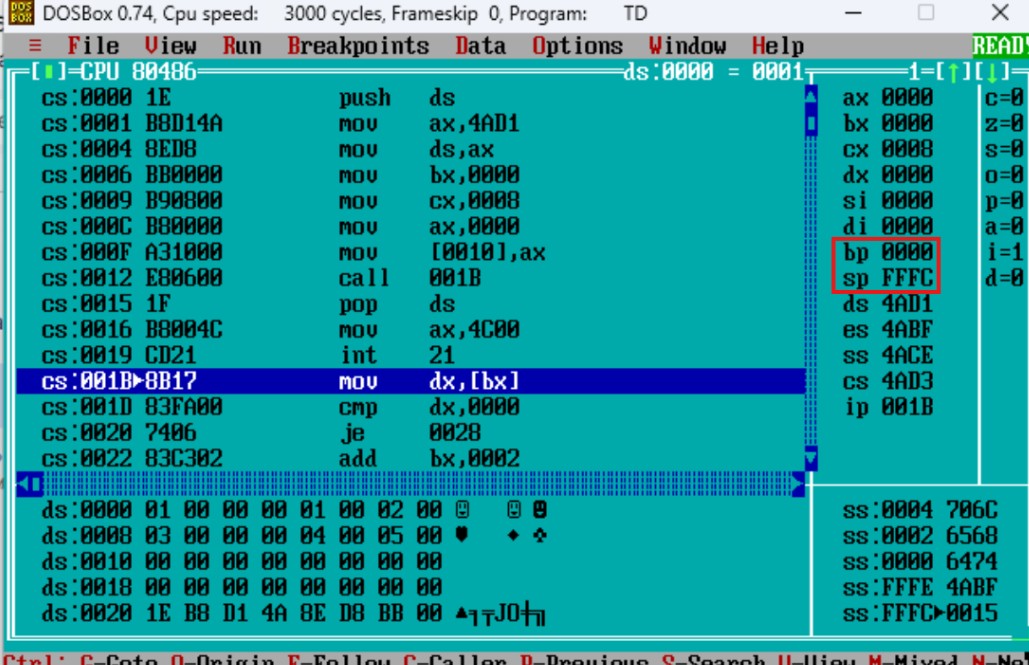
end main

**Отладка**

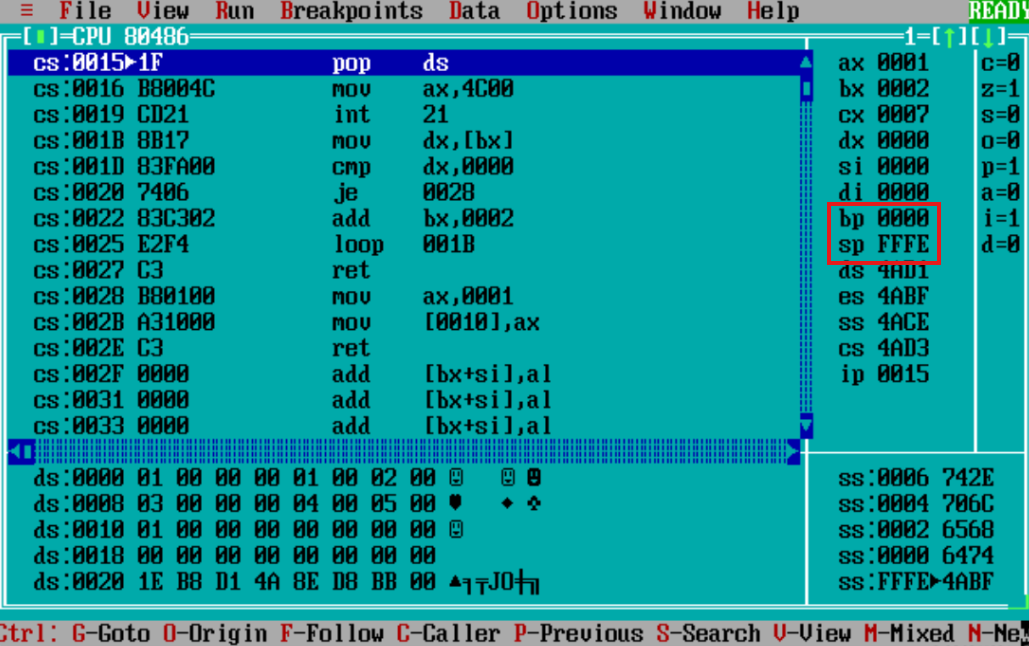
До вызова процедуры



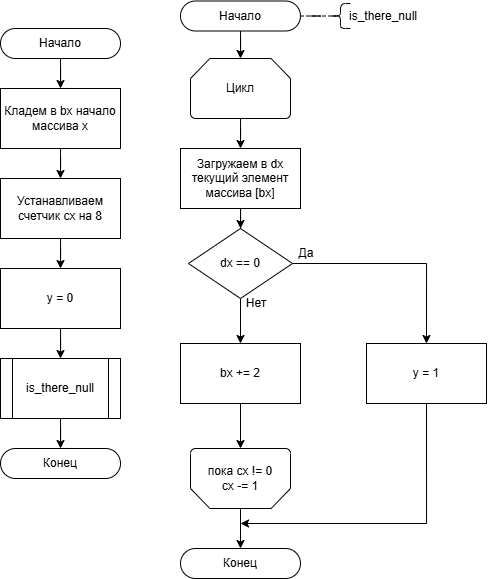
После входа в процедуру



После возврата из процедуры



**Блок-схема**

****

**Задание 2**

**Листинг**

stseg segment para stack

dw 16 dup(?)

stseg ends

dseg segment para

x dw 1, 0, 1, 2, 3, 0, 4, 5

y dw ?

dseg ends

cseg segment para

main proc far

assume cs:cseg, ds:dseg, ss:stseg

push ds

mov ax, 0

push ax

mov ax, dseg

mov ds, ax

mov bx, offset x

mov cx, 8

; Инициализируем y значением 0

mov ax, 0

mov y, ax

; Передаём параметры в стек (адрес массива и его длину)

push bx

push cx

call is\_there\_null

end\_proc:

ret

main endp

is\_there\_null proc near

push bp

mov bp, sp

mov bx, [bp+6] ; Получаем адрес массива

mov cx, [bp+4] ; Получаем длину массива

loop\_start:

mov dx, [bx] ; Загружаем текущий элемент массива

cmp dx, 0

je set\_y\_to\_one ; Если элемент равен 0, переходим к установке y = 1

add bx, 2 ; Переход к следующему элементу

loop loop\_start ; Уменьшаем CX и повторяем цикл, если CX не ноль

jmp exit\_proc ; Выход, если нулевых элементов не найдено

set\_y\_to\_one:

mov ax, 1

mov y, ax ; Присваиваем y значение 1

exit\_proc:

pop bp

ret 4 ; Освобождаем параметры из стека

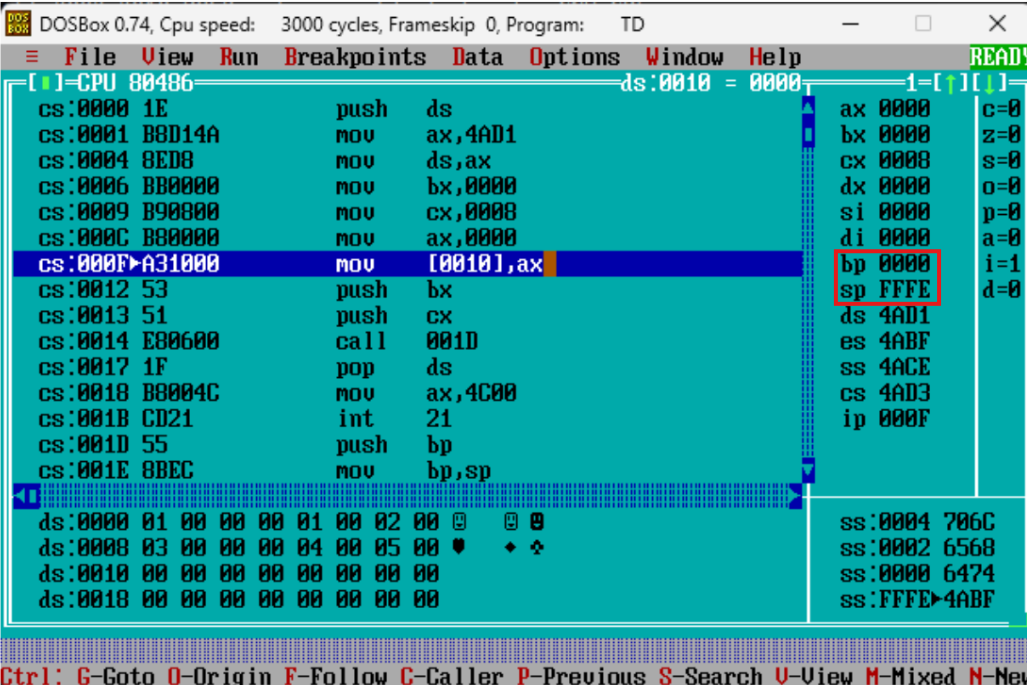
is\_there\_null endp

cseg ends

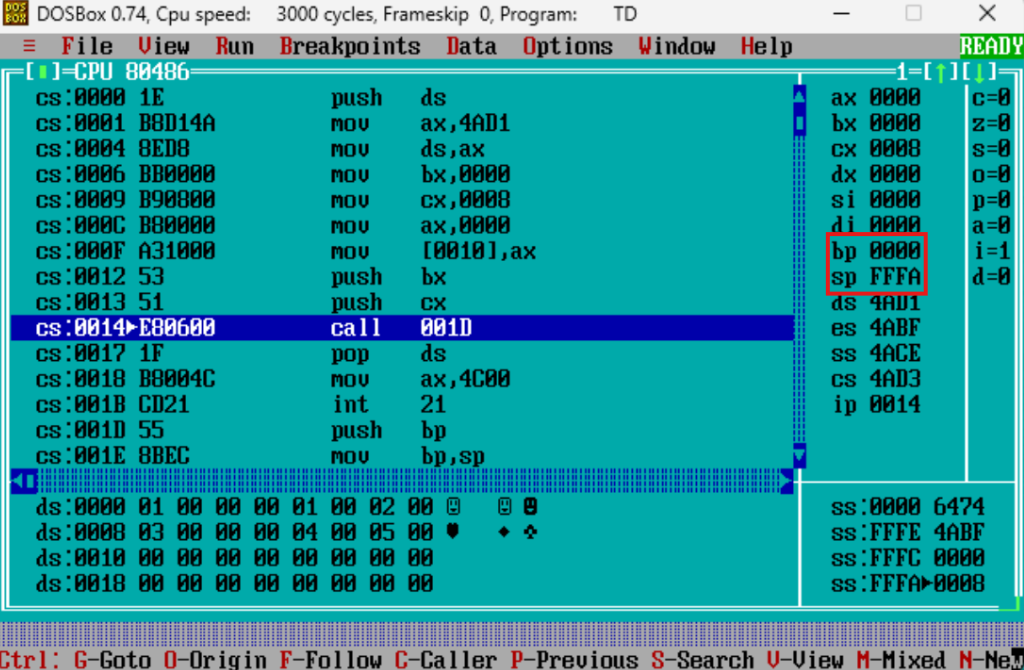
end main

**Отладка**

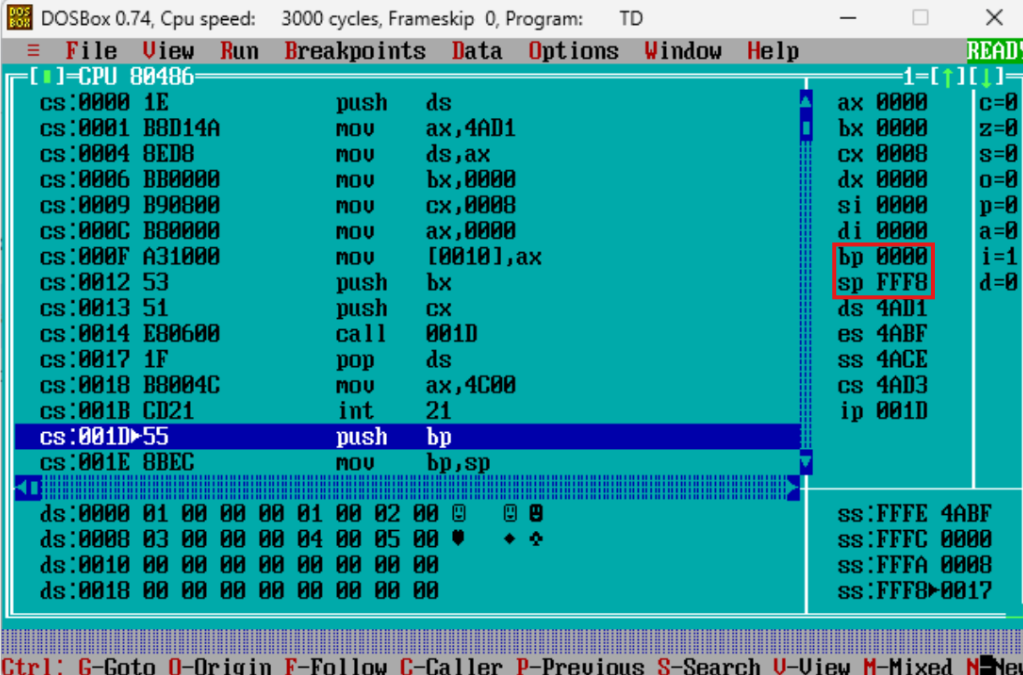
До пуша в стек



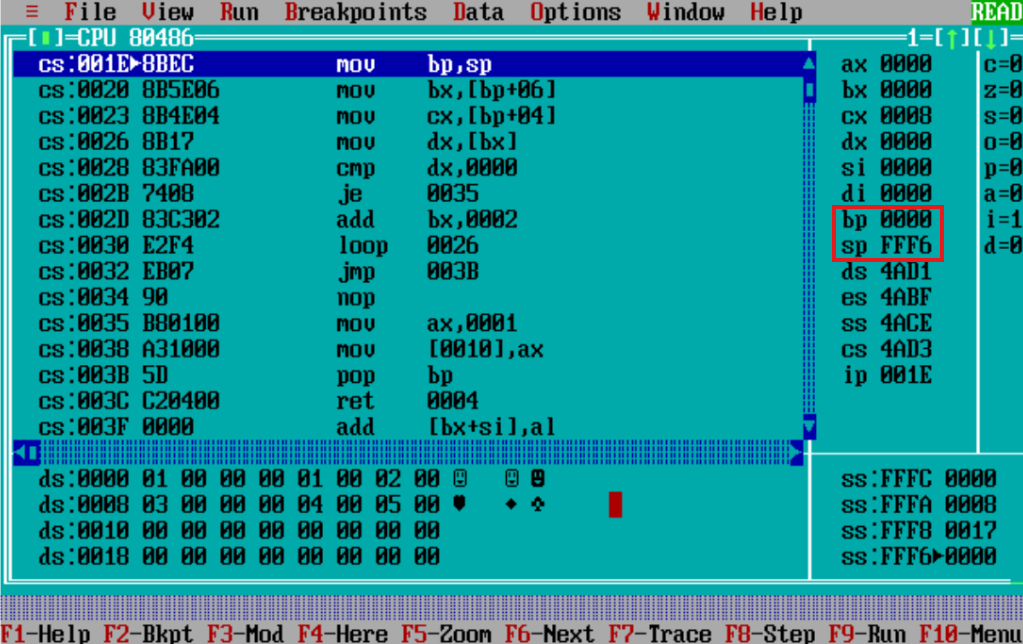
После пуша в стек



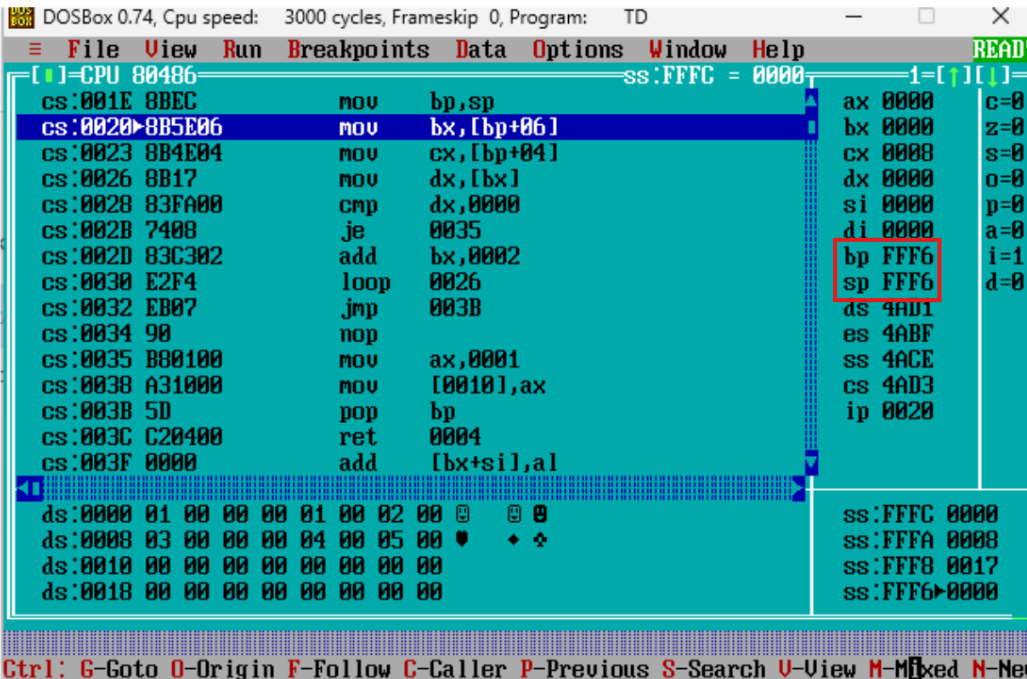
После вызова процедуры



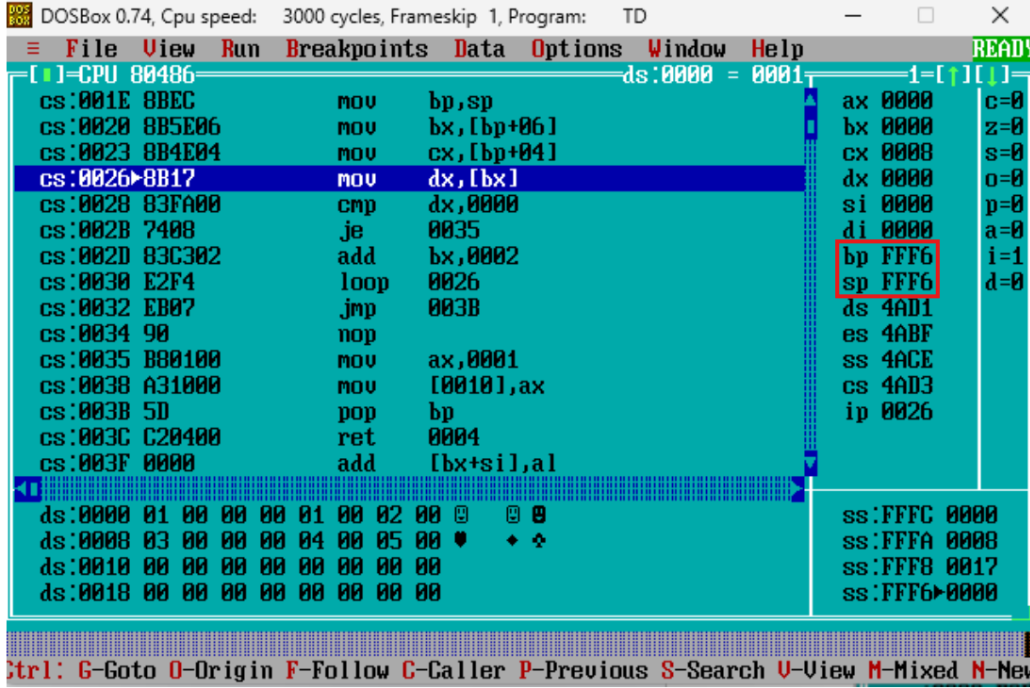
После пуша bp



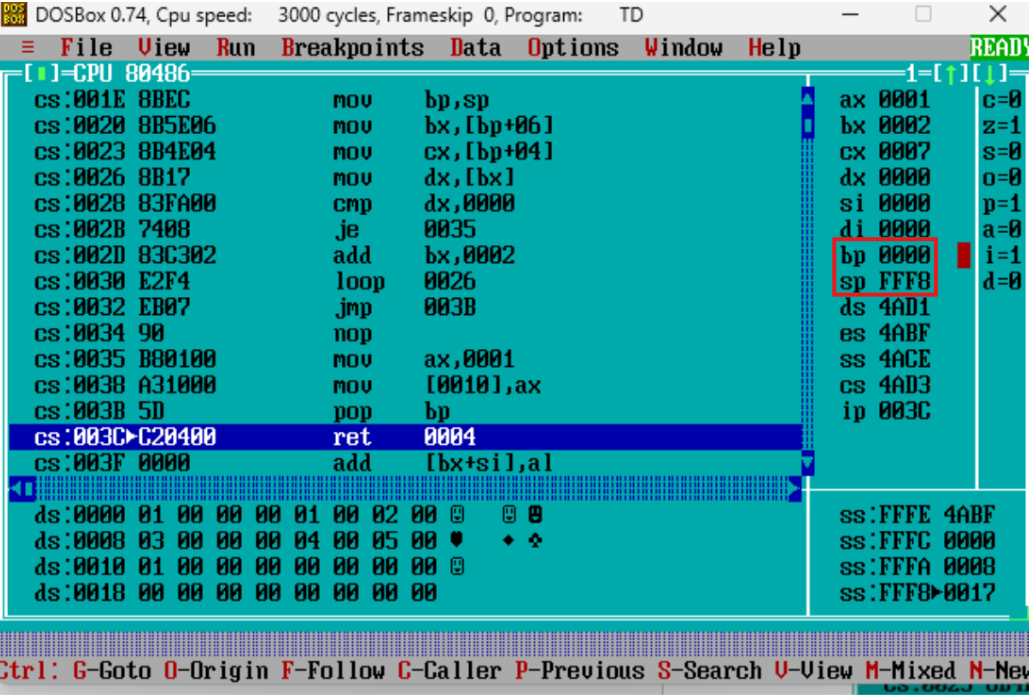
Заносим sp в bp



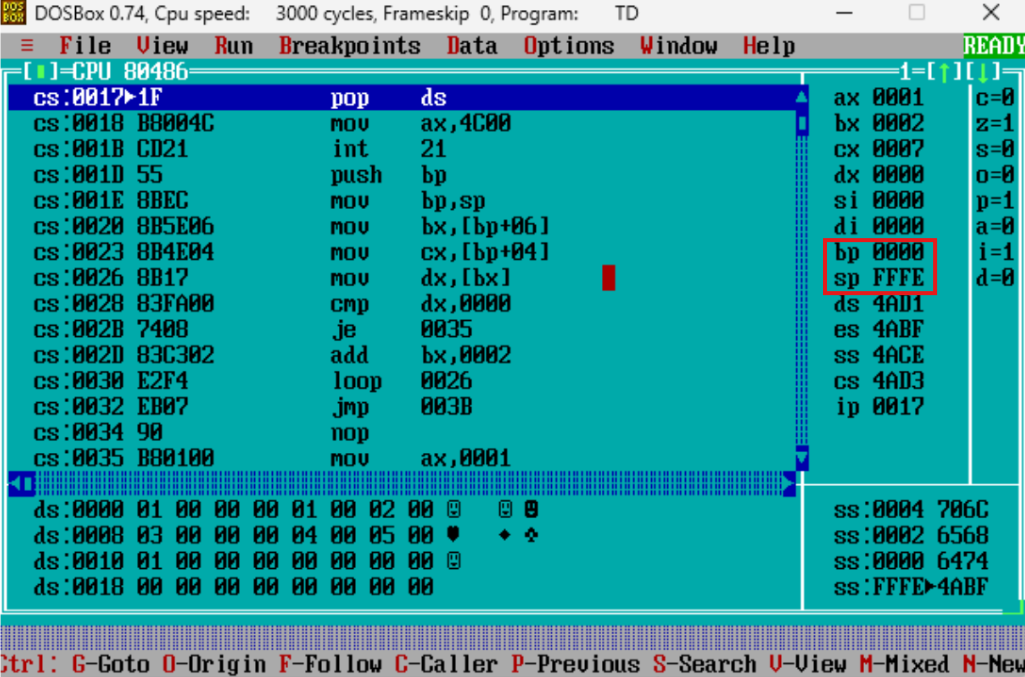
Берем переменные из стека:



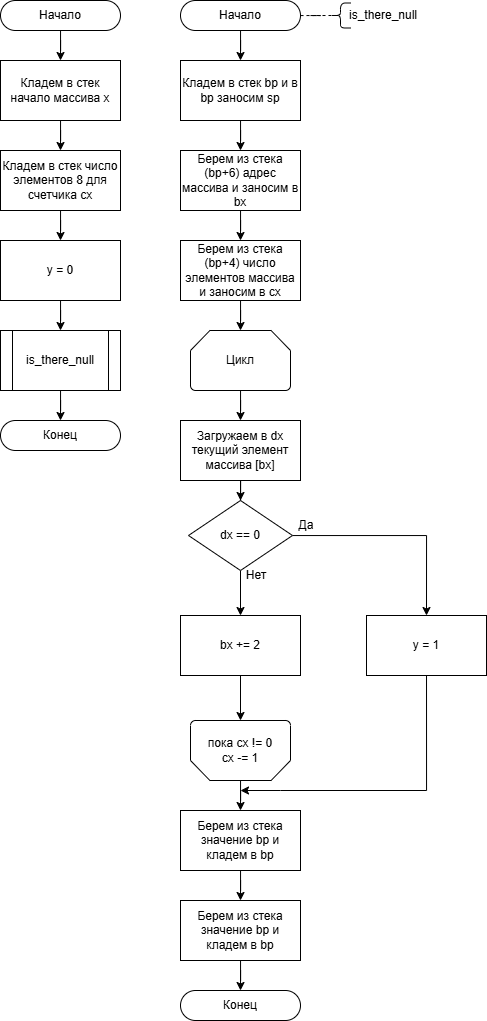
После pop в подпрограмме



Подпрограмма завершилась (Выполнилось ret 4)



**Блок-схема**

****

**Вывод**

Изучили особенности выполнения команд работы с подпрограммами и передачи параметров в подпрограммы.