# **ЛАБОРАТОРНАЯ РАБОТА №3**

**Тема:** «Циклические и разветвляющиеся программы на языке ассемблера».

**Цель работы:** Изучить конструкции циклических и разветвляющихся программ на языке ассемблера, принципы действия и отладки программ на языке ассемблера.

**Ход работы:**

Вариант №4

Я ознакомился с теоретической частью и приступил к выполнению задания. Мне необходимо было разработать программу, которая находит количество положительных чисел из массива байт.

Код:

data segment *; директива начала сегмента данных*

count dw ?

mass db -1,0,2,3,-8,6,-7,-6,9,4 *; массив из 10 элементов, каждый размером в байт*

data ends

code segment ; *директива начала сегмента кодов*

assume cs: code, ds: data *; установка соответсвий сегментов и их адресов*

start: *; метка входа*

mov ax,data *; Загрузка адреса*

mov ds,ax *; сегмента данных*

lea bx,mass *; Загружаем в bx адрес массива*

mov cx,10 *; Установить счетчик повторений цикла*

mov ax,0

beg: cmp byte ptr[bx],0 *; Сравниваем однобайтовое значение по адресу, лежащему в bx, с нулем*

jle no *; если <= 0 то переходим к метке no*

inc ax *; если >0, то увеличиваем количество на единицу*

no:

inc bx *; переход к следующему элементу массива*

loop beg *; повтор, пока cx не станет равным нулю*

mov count, ax *; результат записывается в count*

quit: *; метка выхода*

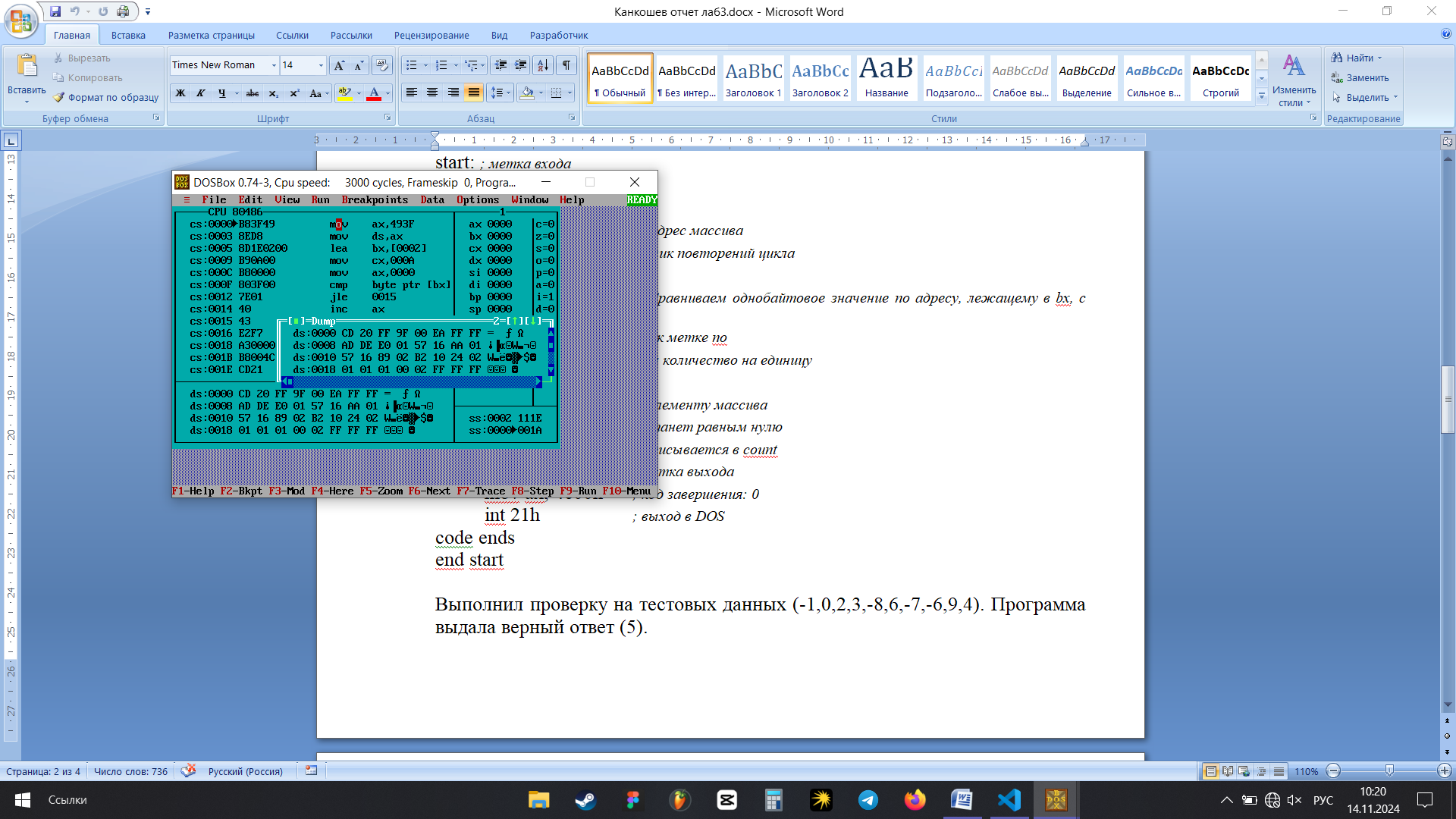
mov ax, 4c00h *; код завершения: 0*

int 21h *; выход в DOS*

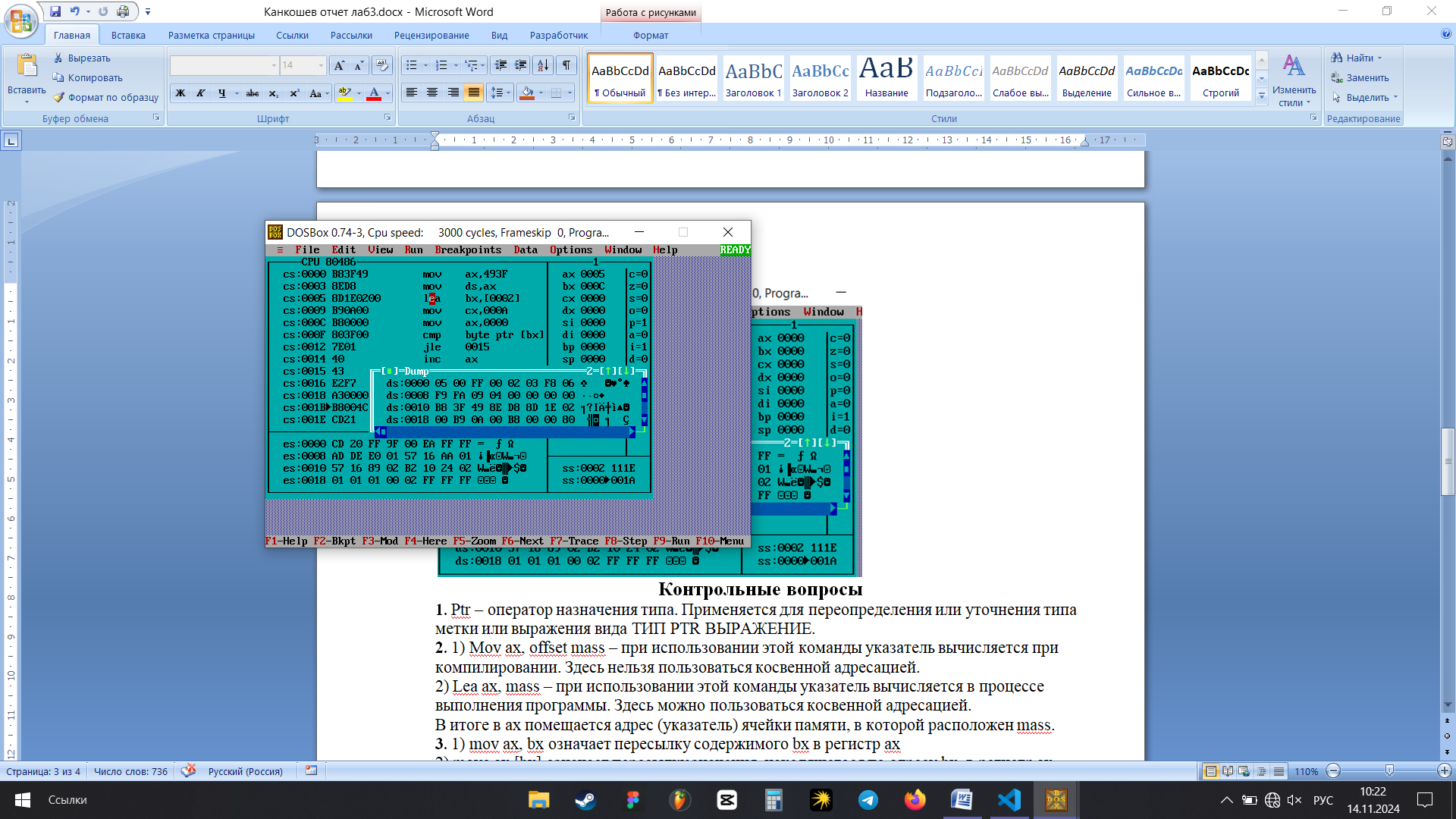
code ends

end start

Выполнил проверку на тестовых данных (-1,0,2,3,-8,6,-7,-6,9,4). Программа выдала верный ответ (5).



Окно dump до начала выполнения программы



Окно dump после выполнения программы

**Контрольные вопросы**

**1.** Ptr – оператор назначения типа. Применяется для переопределения или уточнения типа метки или выражения вида ТИП PTR ВЫРАЖЕНИЕ.

**2.** 1) Mov ax, offset mass – при использовании этой команды указатель вычисляется при компилировании. Здесь нельзя пользоваться косвенной адресацией.

2) Lea ax, mass – при использовании этой команды указатель вычисляется в процессе выполнения программы. Здесь можно пользоваться косвенной адресацией.  
В итоге в ах помещается адрес (указатель) ячейки памяти, в которой расположен mass.

**3.** 1) mov ax, bx означает пересылку содержимого bx в регистр аx

2) mov ax,[bx] означает пересылку значения, находящегося по адресу bx, в регистр аx

**4.** 1) mov ax,[bp] - в ах заносится значение слова памяти, на которое указывает ss:bp (из сегмента стека)

2) mov ax,[bx] - в ах заносится значение слова памяти, на которое указывает ds:bx(из сегмента данных)

**5.** Ничем, после ассемблирования обе будут mov ax, [bx+2]

**6.** Ничем, после ассемблирования обе будут mov ax, [bx+si]

**7.** У jmp есть 3 разновидности: short, near и far

**8.** С помощью безусловного перехода jmp far ptr <операнд> или jmp far ptr <адрес операнда>

**9.** Условные переходы предназначены только для коротких переходов(от -128 до +127 байт ). Для не коротких переходов нужно воспользоваться jmp с любым типом кроме short  
  
Возможный вариант:(сделать 3 прыжка с метки на метку)  
…  
<условный переход> vspomogatelnaya2  
vspomogatelnaya2:  
<условный переход> vspomogatelnaya1  
vspomogatelnaya1:  
<условный переход> glavnaya  
glavnaya:  
…  
  
Либо можно инверсировать условие. Например, вместо a>0, которое находится на расстоянии больше 127 байт, искать a<=0, которое находится на нужном расстоянии. Тогда на нужное нам условие будет переходить при невыполнении a<=0 (Так предложил сам преподаватель)

**10.** Команды loop, loope, loopne необходимы для введения дополнительных логических условий на повторение цикла

Loop -  повторять пока cx не равен нулю

Loope – повторять, пока cx не равно нулю, а zf(флаг нуля) равен 1

Loopne – повторять, пока cx не равно нулю, а zf(флаг нуля) равен 0

**11.** Команда LOOP уменьшает регистр-счетчик без изменения какого-либо из флагов. Затем проверяются условия характерные для конкретной формы используемой команды LOOP. Если условия удовлетворяются, то происходит короткий переход на метку, заданную операндом команды LOOP. Если атрибут размера адреса равен 16 бит, то в качестве счетчика используется регистр CX.

**12.** Можно. В регистр сх заносим счётчик, сравниваем с количеством циклов, которые нужно сделать. Если не равно, то увеличиваем сх на 1 и с помощью jnе переходим на метку, которая символизирует начало цикла  
Пример:  
…  
start:  
mov cx,0 ; cx = 0  
for\_loop: ; метка начала цикла  
 inc cx,1 ; увеличиваем cx на 1  
 cmp cx,10 ; сравниваем с 10(общее кол-во циклов)  
 jne for\_loop ; если не равно, то переходим к метке for\_loop

quit: ; если равно, то заканчиваем цикл  
…

**13.** С помощью команды loop можно организовать цикл for. Если из этого цикла for можно сделать цикл while, то можно. В иных случаях нельзя