**Тема:** «Логические инструкции на языке ассемблера»

**Цель работы:** Целью работы является изучение и применение

логических инструкций в программах на языке ассемблера.

**Ход работы:**

Вариант №4

Я ознакомился с теоретической частью и приступил к выполнению задания. Дан массив из 10 байт. Посчитать количество байт с числом

единиц в байте равным три.

Код:

data segment *; директива начала сегмента данных*

count db 0

res db ?

mass db -0010110b,001011b,0110b,10010110b,-00111100b,0010110b,0010110b,0b,1b,111b *; 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

mov cx,10

xor dx,dx

beg: mov al, [bx]

mov dh, cl *; сохраняем счетчик cl в dh*

mov cx,8 *; устанавливаем счётчик бит в байте*

bit: *; метка проверки бита*

test al,1b *; является ли бит единицей?*

jz no *; если нет, то переход к метке*

inc count *; иначе +1 к кол-ву единиц в байте*

no: *; метка, если бит = 0*

shr al,1 *; сдвиг вправо на один разряд*

loop bit *; повторяем проверку уже следующего бита*

cmp count,3 *; в байте 3 единицы?*

jnz netri *; если нет, то переход в метке*

inc dl *; иначе увеличиваем результат на единицу*

netri: *; метка, если в байте не 3 единицы*

mov count,0 *; обнуляем счётчик единиц в байте*

inc bx *; переходим к следующему байту*

mov cl, dh *; получаем счетчик из dh в cl*

loop beg *; повторяем цикл, пока байты не кончатся*

mov res ,dl *; переношу результат из dl в переменную*

quit: *; метка выхода*

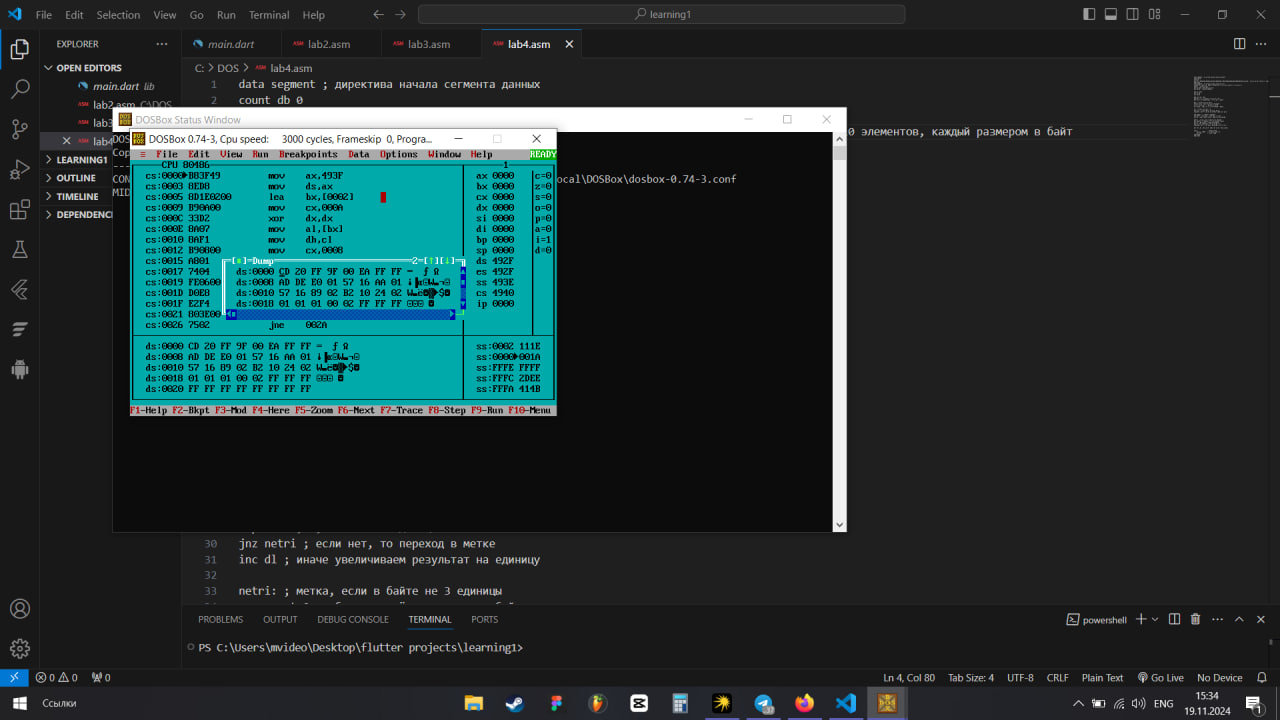
mov ax, 4c00h *; код завершения: 0*

int 21h *; выход в DOS*

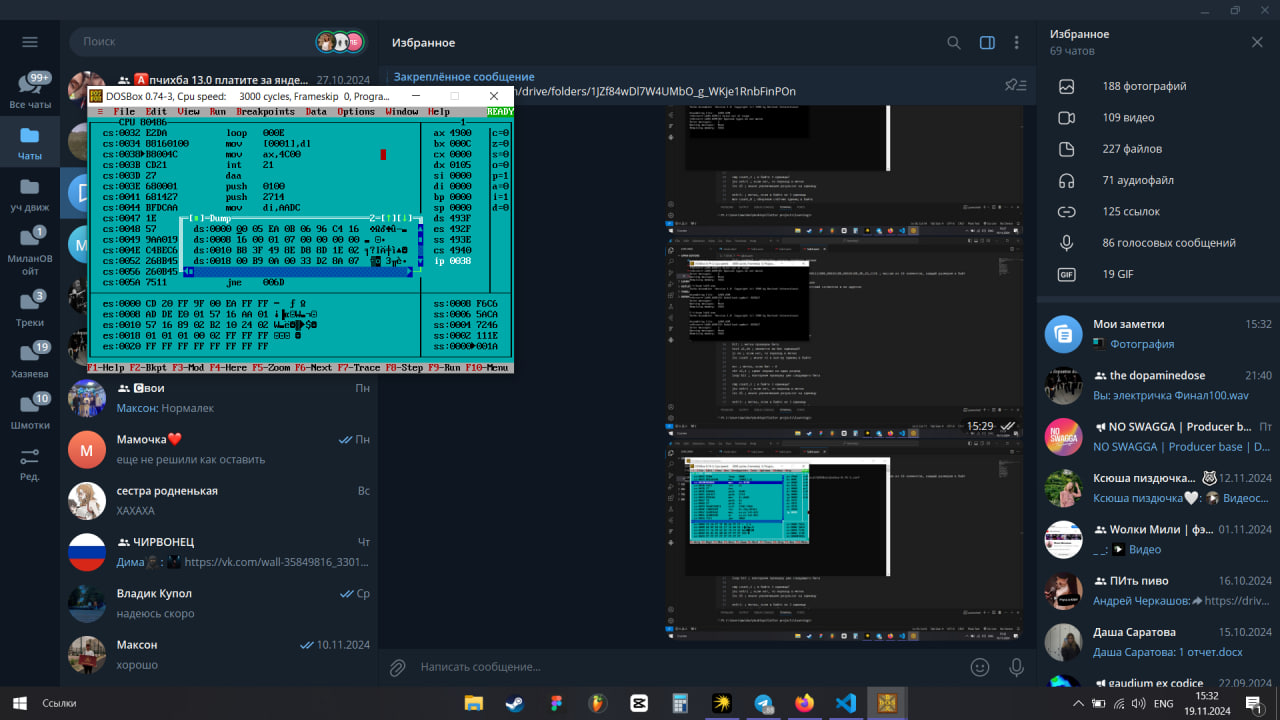
code ends

end start

Выполнил проверку на тестовых данных (-0010110b,001011b,0110b,10010110b,-00111100b,0010110b,0010110b,0b,1b,111b). Программа выдала верный ответ (5).



Окно dump до выполнения программы



Окно dump после выполнения программы

**Контрольные вопросы**

1. Обе команды выполняют логическое И и модифицируют флаги, но

test результат не возвращает, в отличие от and.

2. Чтобы сбросить 5-й бит переменной байта ВВ, нужно написать

команду:

and bb, 11011111b

3. Чтобы установить 5-й бит переменной байта ВВ, нужно написать

команду:

or bb, 00100000b

4. Чтобы инвертировать 5-й бит переменной байта ВВ, нужно написать

команду:

xor bb, 00100000b

5. Чтобы проверить установку 5-ого бита переменной байта ВВ, нужно

написать команду:

test bb, 00100000b

6. Нужно воспользоваться командой test al, 0ffh. Она проверяет

каждый бит и на соответствие побитовой маске(0ff16 = 111111112 ).

Если в числе чётное число единиц, то флаг PF(чётности) становится

равным единице, в противном случае – нулю.

7. Данные инструкции(and, or, xor) всегда сбрасывают флаги

CF(переноса) и OF(переполнения), а также (в зависимости от

результата) изменяют флаги SF(знака), ZF(нуля) и PF(чётности).

8. В логических сдвигах освободившиеся разряды всегда заполняются

нулями, и знаковый бит не требует особого обращения. Вследствие

этого они используются не для обработки арифметической

информации, а для организации более компактной записи некоторых

данных. Выполнение команд логического сдвига не оказывает влияния

на признак результата

9. Команда логического арифметического сдвига определяется как

shl(shr) dst, счетчик, где:

shl(shr) – логический сдвиг влево(вправо);

dst – операнд, в котором будут производиться сдвиги;

счётчик – регистр cl.

Команда сделает столько сдвигов, сколько будет прописано в cl, но

для неё это будет n-ное кол-во повторов строки shl(shr) dst,1 , поэтому

максимальное число двоичных разрядов, на которые можно сдвинуть

операнд равняется единице.