# **ЛАБОРАТОРНАЯ РАБОТА №2**

**Тема:** «Запись программ на языке ассемблера».

**Цель работы:** Изучить конструкции программ на языке ассемблера, принципов действия и отладки программ на языке ассемблера.

**Ход работы:**

Перед выполнением лабораторной работы я еще раз прочитал теоретическую часть и приступил к выполнению задания. Мне необходимо было разработать программу, реализующую формулу 4:

X = - A/2 + 4\*(B+1) + 3C

В результате полный код выглядит следующим образом:

*data segment ;директива начала сегмента данных*

*a dw 00000h*

*b dw 0ffffh*

*c dw 0ffffh*

*x dd ?*

*data ends ; директива конца сегмента данных*

*code segment ; директива начала сегмента кодов*

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

*; cs - указывает на код программы, ds - на данные*

*start:*

*mov ax, data ; загрузка адреса*

*mov ds, ax ; сегмента данных*

*mov ax, a ; заносим в ax значение а*

*mov bx, 00002h*

*div bx ; делим а на 2*

*mov cx, ax ; заносим значение (A/2) в cx*

*mov dx, 00000h*

*mov ax, b ; заносим значение b в ax*

*mov bx, 00004h ; заносим множитель 4 в bx*

*mul bx ; результат (4\*b) заносится в dx:ax*

*add ax, 00004h ; выполняем операцию (4\*b)+4*

*adc dx, 00000h ; выполняем сложение с переносом, чтоб не было неточностей*

*add ax, c ; добавляем значение С*

*adc dx, 00000h ; выполняем сложение с переносом, чтоб не было неточностей*

*add ax, c ; выполняем сложение числа с самим собой (С\*2)*

*adc dx, 00000h ; выполняем сложение с переносом, чтоб не было неточностей*

*add ax, c ; выполняем сложение числа с самим собой (С\*3)*

*adc dx, 00000h ; выполняем сложение с переносом, чтоб не было неточностей*

*sub ax, cx ; выполняем вычитание (4\*(В+1)+3С)-А/2*

*sbb dx, 00000h ; выполняем вычитание с заемом, чтоб не было неточностей*

*mov word ptr [x], ax ; Сохраняем младшие 16 бит*

*mov word ptr [x+2], dx ; Сохраняем старшие 16 бит*

*quit: ; метка выхода*

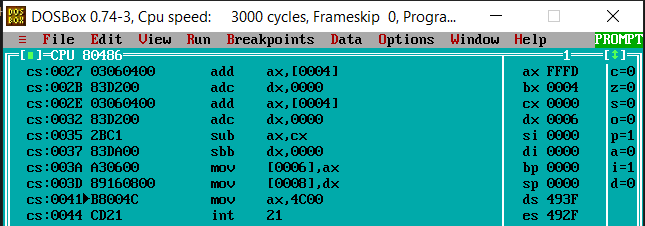
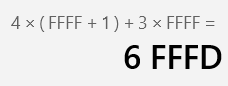
*mov ax, 4c00h ; код завершения: 0*

*int 21h ; выход в DOS*

*code ends*

*end start*

После выполнил ручную проверку на калькуляторе. Результаты совпали.

 dx:ax = 6FFFDh

**Вывод:** В ходе выполнения лабораторной работы я изучил конструкции программ, принципов действия и отладки программ на языке ассемблера.

**Контрольные вопросы.**

**1.** а) Регистровая прямая - операнд находится в регистре;  
б) Непосредственная - непосредственный операнд (константа) присутствует в команде;  
в) Прямая - исполнительный адрес операнда присутствует в команде;  
г) Регистровая косвенная - регистр содержит адрес операнда;  
д) Регистровая относительная - адрес операнда вычисляется как сумма

содержимого регистра и смещения;  
е) Индексно - базовая - адрес операнда вычисляется как сумма

содержимых базового и индексного регистров и смещения.

**2.** Директивой называется команда транслятору для выполнения определённых данной директивой действий, сама директива в текст транслированной программы не включается. В ассемблере доступны директива задания исходных данных и директива использования сегментных регистров по умолчанию,

**3.** Есть 3 типа регистров: общего назначения, сегментные и специальные. Первые используются для хранения данных, вторые – только хранят адреса сегментов в памяти, а каждый из последних имеет свою функцию. SP - указатель стека, BP - указатель базы стека, IP - указатель инструкций, а FLAGS - регистр флагов.

**4.** Регистр флагов – это специальный регистр, содержащий набор битовых флагов, определяющий текущее состояние процессора и результат выполнения предыдущей команды.

**5.** Структура программы на ассемблере.

<имя сегмента> segment ; обычно в этом блоке пишутся данные

команды или директивы

<имя сегмента> ends

<имя сегмента> segment ; обычно в этом блоке пишется код

команды или директивы

<имя сегмента> ends

end <метка входа в программу>

**6.** При обработке программ процесс взаимодействует с ОЗУ или RAM. Состоит из однобайтовых ячеек, обращение к ним происходит по физическим адресам. Число ячеек составляет двойка, возведённая в степень ширины шины адреса. Данные можно читать и сохранять, указывая адрес нужной ячейки.

Сегментация памяти заключается в том, что истинный, физический адрес ячейки хранится в двух регистрах. Один из них – сегментный, он хранит адрес начала блока памяти, который и называется сегментом. Второй регистр хранит величину смещения адреса требуемой ячейки от начала сегмента.

**7.**  Директива Segment отмечает начало сегмента данных/кода.

Директива Ends отмечает конец сегмента данных/кода.

**8.** Директива assume позволяет избежать повторения одной и той же строки в коде, а именно строки задания адреса в память. С помощью assume задаётся по умолчанию сегментный регистр для сегмента кодов, т.е. она указывает ассемблеру с каким сегментом или группой сегментов связан тот или иной сегментный регистр.

**9.** Буква d (define) в db и dw определяет начало массива данных, а сами b и w определяют размер констант, входящих в массив: b – байт, w – слово (два байта).

**10.** Директива DUP используется для заполнения больших массивов. Она дублирует определённое количество раз заданные данные.

**11.** Директива end отмечает конец текста программы и указывает ассемблеру, где завершить трансляцию. Поэтому директива end должна присутствовать в каждой программе.

**12.** Строка программы, в общем случае, состоит из четырех полей:

Поле метки, поле операции, поле операндов и поле комментария.

**13.** Обязательным является поле операции

**14.** 1) mov ax, bx означает пересылку содержимого bx в регистр аx  
2) move ax,[bx] означает пересылку значения, находящегося по адресу bx, в регистр аx

3) move [ax],bx означает пересылку содержимого bx по адресу, находящемуся в регистре аx

**15.** В чем разница между командой mov aх,1 и директивой a dw 1?

Директива a dw 1 задаёт константе а значение 1, которое может равняться слову, а команда mov пересылает данные константы/операнда в другой операнд