Лабораторное занятие
Изучение процесса разработки модулей на языке ассемблера

1 **Цель работы:**

1.1 Приобрести навыки работы с цепочечными командами на ассемблере

2 **Литература:**

2.1 Приложение п.9.

3 **Оборудование:**

3.1 Персональный компьютер;

4 **Подготовка к работе:** 4.1 Изучить приложение

5 **Задание:**

5.1 Выполнить задания из пункта 6.

5.2 Составить электронный отчет.

5.3 Ответить на контрольные вопросы.

6 **Порядок выполнения работы:**

Код для последующих заданий должен быть написан в ассемблерных вставках на masm! Ознакомьтесь с приложением, перед выполнением работы.

6.1 Разработать модуль, содержащий процедуру для сложения двух чисел.

6.2 Добавить в модуль процедуру для возведения числа в указанную степень.

6.3 Добавить в модуль процедуру для перевода из градусов °F в градусы °C.

7 **Содержание отчёта:**

7.1 Титульный лист.

7.2 Цель работы.

7.3 Последовательность действий по выполнению задания.

7.4 Ответить на контрольные вопросы.

7.5 Вывод по проделанной работе.

8 **Контрольные вопросы:**

8.1 Что такое цепочечные команды и для чего они нужны?

8.2 Что такое префикс повторения?

9 **Приложение**

"X:\Абрамова\Системное программирование\Лекции\Модули на ассемблере.pdf"

"X:\Абрамова\Системное программирование\Лекции\Процедуры и функции.pdf"

Пример размещения процедур на ассемблере во внешнем файле и их вызова:

Файл .cpp с функцией main. Здесь начинается выполнение кода. В нем объявляем фнешнюю функции start из модуля на ассемблере и функцию print

для печати на консоль переданного в параметрах числа (понадобится для того чтоб выводить кода на ассемблере значения)

```
// lection.cpp: определяет точку входа для консольного приложени
//

#include "stdafx.h"
#include <iostream>
 using namespace std;

extern "C"{
    void start();
    void print(int a){
        cout << a << endl;
    }
}
int _tmain(int argc, _TCHAR* argv[])
{
    start();
    return 0;
}
```

Модуль на ассемблере, содержащий процедуру start (аналог main). Сначала запускается start, а уже из нее процедура sum для сложения двух чисел. Процедуру sum можно было разместить в этом файле, до процедуры start или после, но я разместила во внешнем модуле, поэтому в секции .DATA я объявляю процедуру sum как внешнюю.

```
.686P
.MODEL FLAT, C
.DATA
extern sum:near
extern print:near
.CODE
;//////////////////////////
start PROC;основная процедура
    push 5; помещаем с стек 5
    push 1; помещаем с стек 1
    call sum;вызов проц start
    ;тело функции

    push eax;помещаем eax в стек
    call print; вызов print для вывода eax на кон
    pop eax;очистка стека
start ENDP
;//////////////////////////
```

Модуль с процедурой sum:

```
.686P
.MODEL FLAT, C
.DATA
.CODE

sum PROC
    ;пролог
    push ebp
    mov ebp, esp
    ;достаем из стека переданные значения
    mov eax, [ebp + 8]; достаем первый параметр в eax
    mov ebx, [ebp + 12]; достаем второй параметр в ebx
    add eax, ebx
    pop ebp;эпилог
    ret;возврат в место вызова

sum endp
END
```