Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

Специальность: Обеспечение информационной безопасности автоматизированных систем

**ОТЧЕТ**

ПО ПРАКТИЧЕСКОЙ РАБОТЕ №4  
*(указать вид (этап) практики)*

*(наименование профессионального модуля\_)*

Выполнил:

обучающийся учебной группы   
№3ОИБАС-618

В. А. Бочкарев

*(подпись) (И.О. Фамилия)*

Преподаватель:

И. В. Сиберев

*(И.О. Фамилия)*

*(оценка) (подпись)*

**Москва – 2020г.**

Цель работы: Научиться работать с \_int8, \_int16, \_int32, \_int64, понять, как будет размещать в памяти и работать с таким типом данных компилятор, у которого максимальный размер регистра 32 бита.

Ход работы:

Описать все непонятные команды и операнды, использованные в коде ассемблерной вставки.

#include <stdafx.h>

#include <iostream>

#include <stdio.h>

#include <ctime>

using namespace std;

int main()

{

\_\_int64 a(3);

\_\_asm {

//mov dword ptr[a], 3 // поместить в переменную а значение 3

//mov dword ptr[ebp - 8], 0 //

}

\_\_int64 b(4);

\_\_asm {

//mov dword ptr[b], 4 // переметить в переменную b значение 4

//mov dword ptr[ebp - 18h], 0

}

\_\_int64 c(0);

\_\_asm {

//xorps xmm0, xmm0 // обитной операция исключающего или над 128 битным регистром с плавающей запятой

//movlpd qword ptr[c], xmm0 // перемещение в переменную с значения из 128 битного регистра

}

//c = a + b;

\_\_asm {

mov eax, dword ptr[a] //помещаем в eax часть переменной а

add eax, dword ptr[b] //добавляем в eax часть переменной b

mov ecx, dword ptr[ebp - 8] //перемещаем в eсx вторую половину а лежащей в стеке

adc ecx, dword ptr[ebp - 18h] // добавляем c переносом к есx вторую половину b лежащей в стеке

mov dword ptr[c], eax // перемещаем в первую половину переменной с результат из eax

mov dword ptr[ebp - 28h], ecx // перемещаем в вторую половину переменной с (место в стеке) результат из ecx

}

cout << c << endl;

system ("pause");

return 0;

Вывод: В ходе практической работы я разобрался и научился работать с \_int64.