**Лабораторная работа № 2**

Тема: Основные арифметические операции над целыми числами.

Интерфейс с языками высокого уровня.

1. Цель работы

Ознакомиться с арифметическими операциями над целочисленными данными, ознакомиться с правилами оформления ассемблерных процедур, в том числе вызываемых из Си-программ.

2. Краткие теоретические сведения

Основные арифметические операции над целыми числами выполняются с помощью следующих команд:

1)Сложение – ADD, ADC. Команда ADD выполняет арифметическое сложение

приемника и источника, помещает сумму в приемник, не изменяя содержимое

источника. Приемник может быть регистром или переменной, источник может быть числом, регистром или переменной, но нельзя использовать переменную одновременно и для источника, и для приемника. Команда ADD никак не различает числа со знаком и без знака, но, употребляя значения флагов CF

(перенос при сложении чисел без знака), OF (перенос при сложении чисел со

знаком) и SF (знак результата), можно использовать ее и для тех, и для других. Команда ADC во всем аналогична ADD, кроме того, что она выполняет арифметическое сложение приемника, источника и флага СF. Пара команд ADD/ADC используется для сложения чисел повышенной точности.

2)Вычитание – SUB, SBB. Команда SUB вычитает источник из приемника и

помещает разность в приемник. Приемник может быть регистром или переменной, источник может быть числом, регистром или переменной, но нельзя использовать переменную одновременно и для источника, и для приемника. Точно так же, как и команда ADD, SUB не делает различий между числами со знаком и без знака, но

флаги позволяют использовать ее как для тех, так и для других. Команда SBB во

всем аналогична SUB, кроме того, что она вычитает из приемника значение

источника и дополнительно вычитает значение флага CF.

Листинг 2.1. Пример Ch02\_01

//------------------------------------------------

// Ch02\_01.cpp

//------------------------------------------------

#include "stdafx.h"

#include <iostream>

using namespace std;

extern "C" int IntegerAddSub\_(int a, int b, int c, int d);

static void PrintResult(const char\* msg, int a, int b, int c, int d, int result)

{

const char nl = '\n';

cout << msg << nl;

cout << "a = " << a << nl;

cout << "b = " << b << nl;

cout << "c = " << c << nl;

cout << "d = " << d << nl;

cout << "результат = " << result << nl;

cout << nl;

}

int main()

{

int a, b, c, d, result;

a = 10; b = 20; c = 30; d = 18;

result = IntegerAddSub\_(a, b, c, d);

PrintResult("Пример 1", a, b, c, d, result);

a = 101; b = 34; c = -190; d = 25;

result = IntegerAddSub\_(a, b, c, d);

PrintResult("Пример 2", a, b, c, d, result);

return 0;

}

;-------------------------------------------------

; Ch02\_01.asm

;-------------------------------------------------

; extern "C" int IntegerAddSub\_(int a, int b, int c, int d);

.code

IntegerAddSub\_ proc

; Вычисление a + b + c - d

mov eax,ecx ;eax = a

add eax,edx ;eax = a + b

add eax,r8d ;eax = a + b + c

sub eax,r9d ;eax = a + b + c – d

ret ;отправка результата вызывающей функции

IntegerAddSub\_ endp

end

Код C++ в листинге 2.1 в основном прост, но содержит несколько строк, требующих пояснений. Оператор #include "stdafx.h" определяет файл заголовка для

конкретного проекта, который содержит ссылки на часто используемые системные элементы.Visual Studio автоматически создает этот файл при создании нового проекта C++. Строка extern "C" int IntegerAddSub\_(int a, int b, int c, int

d) представляет собой оператор объявления, который определяет параметры

и возвращаемое значение для функции на языке ассемблера x86-64 IntegerAddSub\_ (все имена функций на языке ассемблера и общедоступные переменные,использованные в этой книге, содержат подчеркивание в конце для лучшей узнаваемости). Модификатор "C" в этой строке объявления указывает компилятору

C++ использовать именование в стиле C для функции IntegerAddSub\_ вместо расширенного имени C++ (расширенное имя C++ включаетдополнительные символы, которые помогают поддерживать перегрузку функции). Он также уведомляет компилятор об использовании связывания в стиле C для указанной функции.

Функция main в файле C++ содержит код, вызывающий функцию на языке

ассемблера IntegerAddSub\_. Эта функция ожидает четыре аргумента типа int

и возвращает одно значение типа int. Как и многие языки программирования,

Visual C++ использует комбинацию регистров процессора и стека для передачи

значений аргументов функции.В данном примере компилятор C++ генерирует

код, который загружает значения a, b, c и d в регистры ECX, EDX, R8D и R9D соответственно до вызова функции IntegerAddSub\_.

В листинге 2.1 код на языке ассемблера x86-64 следует сразу после функции

C++ main.В первую очередь обратите внимание на строки,начинающиеся с точки с запятой.Это строки комментариев.MASM рассматривает любойтекст, следующий за точкой с запятой, как текст комментария. Оператор .code – это директива MASM, которая определяет начало раздела кода на языке ассемблера.

Директива MASM – это команда, которая указывает ассемблеру, как выполнять

определенные действия. В этой книге вы узнаете, как использовать дополнительные директивы.

Оператор IntegerAddSub\_ proc определяет начало функции на языке ассемблера. В конце листинга 2.1 оператор IntegerAddSub\_ endp определяет конец

функции. Как и строка .code, операторы proc и endp являются не исполняемыми

командами, а директивами ассемблера, которые обозначают начало и конец

функции на языке ассемблера. Завершающий оператор end – это обязательная

директива ассемблера, указывающая на завершение операторов для файла на

языке ассемблера. Ассемблер игнорирует любой текст, который присутствует

после директивы end.

Функция на языке ассемблера IntegerAddSub\_ вычисляет значение выражения

a+b+c–d и возвращает это значение в вызывающую функцию C++. Она начинается с команды mov eax, ecx (Move), которая копирует значение a из ECX в EAX.

Обратите внимание, что команда mov не изменяет содержимое регистра ECX.

После выполнения этой команды mov регистры EAX и ECX содержат значение

a. Команда add eax,edx складывает значения в регистрах EAX и EDX. Затем она

сохраняет сумму (т. е. a+b) в регистре EAX. Как и в случае предыдущей команды

mov, содержимое регистра EDX не меняется после выполнения команды сложения. Следующая команда add eax,r8d вычисляет значение a+b+c. Затем идет

команда sub eax,r9d, которая вычисляет окончательное значение a+b+c–d.

2.1. Простая целочисленная арифметика  45

Функция на языке ассемблера x86-64 должна использовать регистр EAX для

возвращения одного 32-битного целочисленного (или int в C++) значения вызывающей функции. В текущем примере для выполнения этого условия не

требуется никаких дополнительных команд, поскольку EAX уже содержит правильное возвращаемое значение. Последняя команда ret (возврат из процедуры) передает управление обратно вызывающей функции main, которая отображает результат. Ниже показан результат выполнения кода примера Ch02\_01:

Пример 1

a = 10

b = 20

c = 30

d = 18

результат = 42

Пример 2

a = 101

b = 34

c = -190

d = 25

результат = -80