**Лабораторная работа № 5**

Тема: Основные операции сдвига над целыми числами.

Интерфейс с языками высокого уровня.

1. Цель работы

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

2. Краткие теоретические сведения

Пример исходного кода лабораторной работы демонстрирует использование команд shl (логический сдвиг влево) и shr (логический сдвиг вправо). Он также иллюстрирует применение нескольких наиболее часто используемых команд, включая cmp (сравнение), ja (перейти, если выше) и xchg (обмен значениями).

Операции сдвига

extern "C" int IntegerShift\_(unsigned int a, unsigned int count, unsigned int\* a\_shl, unsigned int\* a\_shr);

xchg ecx,edx ; обмен содержимого между ecx и edx

mov eax,edx ;eax = a

shl eax,cl ;eax = a << count;

mov [r8],eax ; сохранение результата

Shr edx,cl ;edx = a >> count

mov [r9],edx ; сохранение результата

mov eax,1 ;возвращаемый код успешного выполненияCh02\_02.cpp

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

// Ch02\_03.cpp

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

#include "stdafx.h"

#include <iostream>

#include <iomanip>

#include <bitset>

using namespace std;

extern "C" int IntegerShift\_(unsigned int a, unsigned int count, unsigned int\* a\_shl,

unsigned int\* a\_shr);

static void PrintResult(const char\* s, int rc, unsigned int a, unsigned int count, unsigned

int a\_shl, unsigned int a\_shr)

{

bitset<32> a\_bs(a);

bitset<32> a\_shl\_bs(a\_shl);

bitset<32> a\_shr\_bs(a\_shr);

const int w = 10;

const char nl = '\n';

cout << s << '\n';

cout << "count =" << setw(w) << count << nl;

cout << "a = " << setw(w) << a << " (0b" << a\_bs << ")" << nl;

if (rc == 0)

cout << "Invalid shift count" << nl;

else

{

cout << "shl = " << setw(w) << a\_shl << " (0b" << a\_shl\_bs << ")" << nl;

cout << "shr = " << setw(w) << a\_shr << " (0b" << a\_shr\_bs << ")" << nl;

}

cout << nl;

}

int main()

{

int rc;

unsigned int a, count, a\_shl, a\_shr;

a = 3119;

count = 6;

rc = IntegerShift\_(a, count, &a\_shl, &a\_shr);

PrintResult("Пример 1", rc, a, count, a\_shl, a\_shr);

a = 0x00800080;

count = 4;

rc = IntegerShift\_(a, count, &a\_shl, &a\_shr);

PrintResult("Пример 2", rc, a, count, a\_shl, a\_shr);

a = 0x80000001;

count = 31;

rc = IntegerShift\_(a, count, &a\_shl, &a\_shr);

PrintResult("Пример 3", rc, a, count, a\_shl, a\_shr);

a = 0x55555555;

count = 32;

rc = IntegerShift\_(a, count, &a\_shl, &a\_shr);

PrintResult("Пример 4", rc, a, count, a\_shl, a\_shr);

return 0;

}

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

; Ch02\_03.asm

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

;

; extern "C" int IntegerShift\_(unsigned int a, unsigned int count, unsigned int\* a\_shl,

unsigned int\* a\_shr);

; Возвращает: 0 = ошибка (count >= 32), 1 = успешное выполнение

;

.code

IntegerShift\_ proc

xor eax,eax ; код возврата в случае ошибки

cmp edx,31 ; сравнение count и 31

ja InvalidCount ; переход, если count > 31

xchg ecx,edx ; обмен содержимого между ecx и edx

mov eax,edx ;eax = a

shl eax,cl ;eax = a << count;

mov [r8],eax ; сохранение результата

shr edx,cl ;edx = a >> count

mov [r9],edx ; сохранение результата

mov eax,1 ;возвращаемый код успешного выполнения

InvalidCount:

ret ; возврат в вызывающую функцию

IntegerShift\_ endp

end

В верхней части кода C++ объявление функции на языке ассемблера x86 IntegerShift\_ несколько отличается от предыдущих примеров тем, что определяет два аргумента-указателя. Указатели нужны этой функции, поскольку ей

необходимо возвращать более одного результата своей вызывающей функции. Другое незначительное отличие состоит в том, что возвращаемое из IntegerShift\_ значение int используется, чтобы указать, является ли допустимым

значение count.Оставшийся код C++ в листинге 2.3 выполняет функцию на языке ассемблера IntegerShift\_ с использованием нескольких тестовых примеров

и отображает результаты.

Код на языке ассемблера функции IntegerShift\_ начинается с команды xor

eax,eax, которая устанавливает регистр EAX в ноль. Это сделано для того, чтобы регистр EAX содержал правильный код возврата, если будет обнаружено

недопустимое значение аргумента count. Следующая команда, cmp edx,31, сравнивает содержимое регистра EDX, который содержит счетчик, с постоянным значением 31. Когда процессор выполняет операцию сравнения, он вычитает второй операнд из первого операнда, устанавливает флаги состояния на основе результата этой операции и отбрасывает результат. Если значение count превышает 31, команда ja InvalidCount выполняет переход к месту в программе, указанному операндом-адресатом. Если вы посмотрите на несколько строк вперед, вы заметите оператор с текстом InvalidCount:. Этот текст называется

меткой. Если условие count > 31 истинно, команда ja InvalidCount передает

управление программой первой команде на языке ассемблера сразу после

метки InvalidCount. Обратите внимание, что эта команда может находиться

в той же или в другой строке, как показано в листинге 2.3.

Команда xchg ecx,edx меняет местами значения в регистрах ECX и EDX. Причина в том,что команды shl и shr должны использовать регистр CL для счетчика сдвигов. Команда mov eax,edx копирует значение a в регистр EAX, а последующая команда shl eax,cl сдвигает это значение влево на количество битов, указанное в регистре CL. Значение 64-битного указателя a\_shr передается в функцию

IntegerShift\_ с использованием регистра R8 (в 64-битном программировании

все указатели 64-битные). Команда mov [r8],eax сохраняет результат операции

сдвига в ячейку памяти, указанную содержимым регистра R8.

Последующая команда shr edx,cl сдвигает значение в регистре EDX (который содержит значение аргумента a) вправо на количество битов, указанное

в регистре CL. Затем этот результат сохраняется в ячейку памяти, на которую

указывает регистр R9, в свою очередь содержащий указатель на ячейку памяти, заданную a\_shr.Команда shr используется в функции IntegerShift\_, поскольку аргумент a объявлен как unsigned int. Если a объявить как int, команду sar (арифметический сдвиг вправо) можно использовать для сохранения знакового бита исходного операнда. Команда mov eax,1 загружает в EAX константу, чтобы указать, что значение аргумента count было допустимым. Следует отметить, что проверка count на превышение значения 31 была реализована, чтобы проиллюстрировать проверку аргументов в функции на языке ассемблера. Для команд сдвига, которые используют прямой или переменный счетчик битового сдвига, процессор выполняет операцию маскирования, которая ограничивает отсчет сдвига значением от 0 до 31, когда целевой операнд 32-битный (для 64-битовых операндов используются граничные значения 0 и 63). Вот результат выполнения кода примера Ch02\_03:

Пример 1

count = 6

a = 3119 (0b00000000000000000000110000101111)

shl = 199616 (0b00000000000000110000101111000000)

shr = 48 (0b00000000000000000000000000110000)

Пример 2

count = 4

a = 8388736 (0b00000000100000000000000010000000)

shl = 134219776 (0b00001000000000000000100000000000)

shr = 524296 (0b00000000000010000000000000001000)

Пример 3

count = 31

a = 2147483649 (0b10000000000000000000000000000001)

shl = 2147483648 (0b10000000000000000000000000000000)

shr = 1 (0b00000000000000000000000000000001)

Пример 4

count = 32

a = 1431655765 (0b01010101010101010101010101010101)

Invalid shift count