ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
ПО ЛАБОРАТНОРНОЙ РАБОТЕ № 4

# «Логические команды и команды манипулирования битами»

По дисциплине «Машинно-ориентированные языки программирования»  
Вариант 5

**Выполнил: ст. гр. ТКИ – 341**

Боди Итшан

**Проверил: доц.**

Логинова Л. Н.

**Москва 2024**

**Цель работы**

Изучение команд передачи управления и получение навыка работы с ними.

**Постановка задачи**

1. Занести в память 32 разрядное шестнадцатеричное число в соответствии с вариантом из таблицы 6;
2. Подсчитать количество нулей и единиц в данном числе двумя разными способами;
3. Подсчитать количество парных нулей и парных единиц в данном числе;
4. В младшем байте числа обменять между собой биты 0-7, 1-6, 2-5, 3-4.

|  |  |
| --- | --- |
| Вар. | Число |
| 5 | 12546FD1 |

Ниже приведен код с ассемблерными вставками для выполнения заданий по лабораторной работе

#include <iostream>

int countBitsShiftMethod() {

int count0;

int count1;

\_\_asm {

pushad

mov eax, 0x12546FD1

xor ebx, ebx

start\_loop:

cmp eax, 0

je end\_loop

test eax, 1

jnz one\_bits

shr eax, 1

jmp start\_loop

one\_bits:

inc ebx

shr eax, 1

jmp start\_loop

end\_loop:

mov count1, ebx

mov count0, 32

sub count0, ebx

popad

}

std::cout << "Shift Method: The number of bits of 0 and 1 are respectively "

<< count0 << " and " << count1 << std::endl;

}

int countBitsBSFMethod() {

int count0;

int count1;

\_\_asm {

pushad

mov eax, 0x12546FD1

xor esi, esi

xor ecx, ecx

start\_loop:

bsf ecx, eax

jz end\_loop

inc esi

btr eax, ecx

jmp start\_loop

end\_loop:

mov count1, esi

mov count0, 32

sub count0, esi

popad

}

std::cout << "BSF Method: The number of bits of 0 and 1 are respectively "

<< count0 << " and " << count1 << std::endl;

}

int countPairedBits() {

int count00;

int count11;

\_\_asm {

pushad

mov eax, 0x12546FD1

xor ecx, ecx

xor edx, edx

mov ebx, 31

loop\_start:

shr eax, 1

jc check\_11

test eax, 1

jnz next\_bit

inc ecx

jmp next\_bit

check\_11:

test eax, 1

jz next\_bit

inc edx

next\_bit:

dec ebx

jnz loop\_start

mov count00, ecx

mov count11, edx

popad

}

std::cout << "Count of 00 pairs: " << count00 << std::endl;

std::cout << "Count of 11 pairs: " << count11 << std::endl;

}

int exchangeBits() {

int value;

\_\_asm {

pushad

mov eax, 0x12546FD1

mov ebx, eax

and ebx, 0xFF

; Bits 0 and 7

mov ecx, ebx

and ecx, 0x01

shl ecx, 7

mov edx, ebx

and edx, 0x80

shr edx, 7

and ebx, 0x7E

or ebx, ecx

or ebx, edx

; Bits 1 and 6

mov ecx, ebx

and ecx, 0x02

shl ecx, 5

mov edx, ebx

and edx, 0x40

shr edx, 5

and ebx, 0xBD

or ebx, ecx

or ebx, edx

; Bits 2 and 5

mov ecx, ebx

and ecx, 0x04

shl ecx, 3

mov edx, ebx

and edx, 0x20

shr edx, 3

and ebx, 0xDB

or ebx, ecx

or ebx, edx

; Bits 3 and 4

mov ecx, ebx

and ecx, 0x08

shl ecx, 1

mov edx, ebx

and edx, 0x10

shr edx, 1

and ebx, 0xE7

or ebx, ecx

or ebx, edx

and eax, 0xFFFFFF00

or eax, ebx

mov value, eax

popad

}

std::cout << "Value after direct bit exchange: 0x" << std::hex << value << std::endl;

return value;

}

int main() {

countBitsShiftMethod();

countBitsBSFMethod();

countPairedBits();

exchangeBits();

return 0;

}

**Рисунки, иллюстрирующие работу кода**

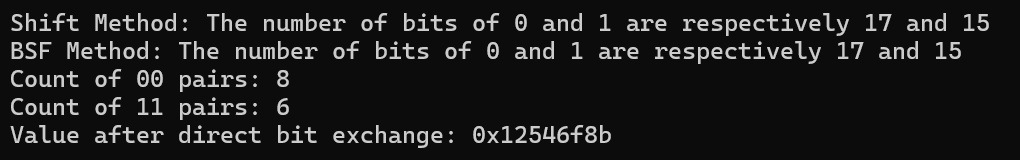


рис.1 - выход из компилятора после выполнения программы

12546FD1 в бинарном виде дает 00010010010101000110111111010001

Количество 0: 17

Количество 1: 15

Теперь давайте посчитаем количество парных нулей и парных единиц в данном числе

Количество 00: 8

Количество 11: 6

Поменяем местами биты 0-7, 1-6, 2-5, 3-4 в младшем байте числа.

Получим 00010010010101000110111110001011, что в шестнадцатеричном виде дает 12546F8B.

**Вывод**

В ходе выполнения данной лабораторной работы были приобретены и закреплены навыки работы с языком ассемблера, в частности с командами, такими как MOV, XOR, SHR, BSF, а также с операциями, такими как тестирование битов и обмен их позиций. Работа с регистрами общего назначения позволила изучить способы обработки битов на низком уровне, а также методы манипулирования данными непосредственно в памяти. Также были исследованы различные подходы для подсчёта количества установленных битов и обмена битами в числе, что позволило углубить понимание bitwise-операций и их применения в задачах с оптимизацией.